UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Universitat Rovira i Virgili
Facultat de Lletres
Departament de Filologies Romàniques

Communication in Membrane Systems with Symbol Objects

PhD Dissertation

Artiom Alhazov

Supervisors:

Prof. Rudolf Freund
Faculty of Informatics
Vienna University of Technology, Austria

Prof. Yurii Rogozhin
Institute of Mathematics and Computer Science
Academy of Sciences of Moldova

Tarragona, Spain, 2006

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

0.1 Abstract

This thesis deals with membrane systems with symbol objects as a theoretical framework of distributed parallel multiset processing systems.

A halting computation can accept, generate or process a number, a vector or a word, so the system globally defines (by the results of all its computations) a set of numbers or a set of vectors or a set of words, (i.e., a language), or a function. The ability of these systems to solve particular problems is investigated, as well as their computational power, e.g., the language families defined by different classes of these systems are compared to the *classical* ones, i.e., regular, context-free, languages generated by extended tabled $0L$ systems, languages generated by matrix grammars without appearance checking, recursively enumerable languages, etc. Special attention is paid to communication of objects between the regions and to the ways of cooperation between the objects.

An attempt to formalize the membrane systems is made, and a software tool is constructed for the non-distributed cooperative variant, the *configuration browser*, i.e., a simulator, where the user chooses the next configuration among the possible ones and can go back. Different distributed models are considered. In the evolution–communication model rewriting-like rules are separated from transport rules. Proton pumping systems are a variant of the evolution–communication systems with a restricted way of cooperation. A special membrane computing model is a purely communicative one: the objects are moved together through a membrane.

Determinism is a special property of computational systems; the question of whether this restriction reduces the computational power is addressed. The results on proton pumping systems can be carried over to the systems with bistable catalysts. Some particular examples of membrane systems applications are solving NP-complete problems in polynomial time, and solving the sorting problem.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Acknowledgements

I would like to thank my family for making this work possible and for their precious support, in particular, my parents for sharing their academic experience. I would like to express my gratitude to all my co-authors for their ideas, efforts and time spent on producing and describing results we have obtained. I thank Chişinău, Tarragona and Vienna for a working environment, and computer technology to facilitate my research in Theoretical Computer Science.

The author[1,2] is very much indebted to Dr. Gheorghe Păun for research experience and writing skills. Many thanks are addressed to the scientific advisers, Dr. Yurii Rogozhin, who brought me to the field of theoretical computer science, and Dr. Rudolf Freund, who made it possible my visit to Vienna to be extremely fruitful, for their expertise and patience, not to mention the proofreading of this work.

A special acknowledgment is due to Prof. Carlos Martín-Vide for encouraging the author's efforts, to the management of IMI for their assistance throughout my PhD program, and to the present and former members and visitors of GRLMC and IMI, as well as the entire membrane systems community, for their moral support and some long scientific discussions, as well as for not letting me concentrate on science too much. Best regards to Dragoş Sburlan, Matteo Cavaliere and Leonor Becerra-Bonache, who have invited me to their defense.

I cannot forget the professors and colleagues of the State University of Moldova and of my School, as they have contributed a lot to my mathematical education, and helped me become a better person.

This work was possible thanks to the financial support of research grant 2001CAJAL-BURV4, as well as projects TIC2002-04220-C03-02 and TIC2003-09319-C03-01 from Rovira i Virgili University.

[1]Research Group on Mathematical Linguistics (GRLMC)
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: artiome.alhazov@estudiants.urv.cat

[2]Institute of Mathematics and Computer Science (IMI)
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: artiom@math.md

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Contents

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 1

Introduction

1.1 About Membrane Computing

Membrane systems, also called P systems, are a framework of distributed parallel computing models, inspired by some basic features of biological membranes. In membrane systems, objects are placed in regions, defined by a membrane structure, and are evolving by means of "reaction rules", associated with regions or with membranes. The rules are applied non-deterministically, and (in most models) in a maximally parallel manner (no rules are applicable to the remaining objects). Objects can also pass through membranes. Many other features are considered. These notions are used to define the transitions between the configurations of the membrane system, and its evolution is used to define the computation.

Three ways of computing were studied: computing functions (data processing), generating and accepting sets (of strings, numbers or vectors). Many different classes of P systems have been investigated, and most of them turned out to be computationally complete with respect to the Turing-Church thesis (i.e., equal in power to Turing machines).

It is quite convenient for a researcher in the domain that a comprehensive bibliography of membrane computing (over 600 titles) can be found on the P systems web page, [205]. There one can also find many articles available for download, including pre-print versions and preliminary proceedings of the meetings, see Table 1.1 (journal articles are sometimes made available online by the publisher, for the subscribed users). The list of author's articles and links to full articles, abstracts and/or publisher pages can be found at

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

http://www.geocities.com/aartiom/pub_aa.html

Meeting	First time	References
Workshop on Membrane Computing	2000	[54], [144], [174], [27], [146], [86]
Brainstorming Week on Membrane Computing	2003	[63], [167], [108]
Brainstorming Workshop on Uncertainty in Membrane Computing	2004	
ESF Exploratory Workshop on Cellular Computing (Complexity Aspects)	2005	[103]
Workshop on Theory and Application of P Systems	2005	[70]

Table 1.1: Membrane Computing Meetings (downloadable from [205])

1.2 About the Topic of This Thesis

This thesis presents studies and a survey of P systems with communicative rules associated to membranes, i.e., evolution–communication P systems, proton pumping P systems and symport / antiport P systems.

The questions considered include (but are not limited to):

- What restrictions can be placed on a model/variant such that it is still computationally complete?
- What is the computational power of a given computing model with certain restrictions (e.g., when maximally parallel object cooperation is not enough for the computational completeness)?
- Can particular problems be solved in a polynomial number of steps by some model with certain restrictions (or what is the time complexity of some problem depending on the features and restrictions of the model)?

The most typical restrictions are

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- restricting the descriptional complexity (e.g., the number of membranes or the number of objects);
- restricting the way of object interaction (e.g., the number of objects involved in a rule, or the way in which they are involved);
- distinguishing a subset of objects with restricted features (e.g., catalysts, bi-stable catalysts, protons);
- considering P systems with some property (e.g., determinism, confluence, ultimate confluence, always halting).

The methods typically used are comparison with (e.g., simulation of) known computational devices (finite automata, context-free grammars, regulated grammars, register machines, 0L systems, grammar systems, other classes of P systems).

1.3 About the Structure of This Thesis

Chapter 1 briefly talks about membrane computing in general, and about the topic of this thesis and its structure in particular. Chapter 2 introduces the relevant background information in Computability (Grammars and Automata Theory). Chapter 3 presents preliminaries of P systems: P systems description, computing with P systems, and the important properties of some P systems. The author's reflections on this are also given. Finally, an attempt to relate the notion of maximal parallelism to a well-studied problem of multi-criterial optimization is stated.

In Chapter 4 the evolution–communication model is formally introduced, and the obtained results are listed. Both generating and accepting cases are investigated, as well as a few restrictions, such as determinism and time-freeness. A special variant is introduced, called proton pumping P systems, and the corresponding (somewhat surprising) results are presented.

In Chapter 5 we introduce P systems having only symport / antiport rules, stating the studies of the power of rules moving a small number of objects between regions. The history of the research and the latest results are presented, both on cell-like P systems and tissue P systems.

Chapter 6 considers the same model as Chapter 5, but gives answers to the question of what can be computed by symport / antiport P systems with

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

a small alphabet of objects. Again, both cell-like P systems and tissue P systems are considered.

Chapter 7 dwells upon some applications of P systems. An application of evolution–communication P systems in the Sorting Theory is presented in Section 7.1, an application of symport / antiport P systems with membrane division in Complexity Theory is presented in Section 7.2, and an application of proton pumping P systems in Multiset Rewriting Theory is presented in Section 7.3 by relating protons to bi-stable catalysts.

Chapter 8 outlines the results reflected in this thesis and makes a number of remarks, as well as some interesting open questions.

All results presented in Chapters 4, 5, 6, 7 are original results obtained by the author himself or in cooperation with some co-authors.[1]

The last section of Chapter 3 is based on article [4]. Chapter 4 is based on articles [5], [10], [7], [11] and [6]. Chapter 5 is based on articles [24], [31], [22], [30] and [21]. Chapter 6 is based on articles [13], [16], [17] and [15]. Chapter 7 is based on articles [32], [33], [9] and [6].

This thesis is intended to give a complete picture of investigations in membrane systems related to communication (rules of moving objects assigned to membranes).

[1]For some results (e.g., Theorem 4.4.1) a new proof was constructed, considering the studies of the field after the first proof. Some other results (e.g., Theorem 4.4.2) are previously unpublished, and are made for obtaining a complete picture.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 2

Prerequisites

This chapter includes a summary of Set Theory, Algebra, Formal Language Theory and Theoretical Computer Science topics related to membrane systems and to this thesis in particular.

2.1 Basic Prerequisites

2.1.1 Set Theory and Algebra

Let $\mathbb{N}$ be the set of natural numbers (non-negative integers). An (m-dimensional) vector (of non-negative integers) is an m-tuple $\vec{v} = (a_1, \cdots, a_m) = (a_i)_{1 \leq i \leq m}$. It can be given as $v : \{1, \cdots, m\} \to \mathbb{N}$.

Consider a finite set V. A (finite) *multiset* over V is a mapping $M : V \to \mathbb{N}$. For $a \in V$, the number $M(a)$ is called the multiplicity of a in M. If V is finite, ordered and fixed, then the set V^o of multisets over V (also denoted as $\mathbb{N}^V$) is isomorphic to the set $\mathbb{N}^{|V|}$ of $|V|$-dimensional vectors of natural numbers (a multiset M over $V = \{a_1, \cdots, a_k\}$ can be represented by a vector $(M(a_1), \cdots, M(a_k)) \in \mathbb{N}^k$).

Usually, in this thesis (and in P system area), multisets are represented by strings, e.g., M can be represented by $w = \prod_{i=1}^{k} ({a_i}^{M(a_i)})$ (or by any permutation of w since the order of the symbols is not important): the multiplicity $M(a)$ of each symbol a is represented by the number $|w|_a$ of occurrences of the symbol a in w.

A *multiset-rewriting system* is defined as a tuple $G = (V, R, w)$ where V is the alphabet, R is a finite set of rules of the form $r : u \to v$, $u \in V^+$, $v \in V^*$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

r is called the label of the rule, and $w \in V^*$ is the initial configuration. The label uniquely identifies the rule and the set of all labels is denoted by $Lab(R)$.

A *graph* (directed, without loops) is a construct $G = (V, U)$, where V is a (finite) set of vertices (also called nodes) and $U \subseteq V \times V - \{(x, x) \mid x \in V\}$ are called edges. An edge $e = (x, y)$ is said to be from x to y. The edge e is also called *adjacent* to x and to y, and x and y are called *incident*.

A subgraph $G_A = (A \subseteq V, U' \subseteq U)$ of a graph $G = (V, U)$ is called *generated* by A if $U' = U \cap A \times A$. A graph is called *complete* if any two (different) nodes are incident.

By 2^M we will denote the *powerset* (the set of all subsets) of M.

A *semigroup* $(M, *)$ is a set M on which a *binary operation* (a function $* : M \times M \to M$) is introduced, which is *associative* $(x * (y * z) = (x * y) * z$ $\forall x, y, z \in M)$.

A *monoid* $(M, *)$ is a semigroup with a *neutral element* $(\exists e \in M\colon\ x * e = e * x = x\ \forall x \in M)$.

A *group* $(M, *)$ is a monoid, where any $x \in M$ has a *symmetrical* element $y \in M$ $(x * y = y * x = e)$

A semigroup, monoid or a group is called *abelian* (commutative) if the operation is commutative $(x * y = y * x\ \forall x, y \in M)$

A monoid $(M, *)$ is *generated* by V by the operation $*$ if it is the smallest monoid containing V.

A monoid $(M, *)$ generated by V is *free* if every element of M can be represented as a product of a finite number of elements of V in a unique way.

2.1.2 Languages

An *alphabet* is a finite non-empty set of abstract symbols.

For an alphabet V the *universal language* V^* is a set of all strings of symbols of V. Notice that V^* is a free monoid, generated from V by the concatenation operation. (The neutral element of this monoid is the empty string, denoted by λ.) The set $V^* - \{\lambda\}$ of non-empty strings over V is denoted by V^+.

A *formal language* L is a subset of a universal language.

The number of occurrences of a given symbol $a \in V$ in $x \in V^*$ is denoted by $|x|_a$. The number of occurrences of symbols from $U \subseteq V$ in $x \in V^*$ is $|x|_U = \sum_{a \in U} |x|_a$. The *length* of a string $x \in V^*$ is $|x| = |x|_V$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The *length set* of a language L is $N(L) = \{|x| \mid x \in L\}$. For a family of languages $F \subseteq 2^{V^*}$, NF stands for the family of length sets of languages in F.

For $k \geq 1$, by N_kF we denote the family of length sets of such languages $L \in F$ that $n \geq k$ for every $n \in N(L)$.

The *Parikh vector* of a string $x \in V^*$ with respect to the (ordered) alphabet $V = \{a_i \mid 1 \leq i \leq n\}$ is $\Psi_V(x) = (|x|_{a_i})_{1 \leq i \leq n}$.

The *Parikh mapping* associated with V is the function $\Psi_V : 2^{V^*} \to 2^{\mathbb{N}^{|V|}}$ defined by $\Psi_V(L) = \{\Psi_V(x) \mid x \in L\}$.

If $F \subseteq 2^{V^*}$ (F is a family of languages), then by PsF we denote the family of Parikh images of languages in F.

If we call the words, differing only in the order of symbols, equivalent, then the equivalence classes can be described by the number of occurrences of each symbol, i.e., by their Parikh set, and vice-versa. The order of symbols in this case is irrelevant, so the operation (concatenation of string representatives, or union of multisets, or addition of vectors) is commutative. This is why the set of equivalence classes is called a *commutative monoid*, generated from the alphabet, and its subsets are called *commutative languages*.

Given a word $w \in V^*$, we denote by $Sub(w)$ the set $\{y \in V^* \mid w = xyz\}$ of all subwords of w, and by $alph(w)$ the set $\{a \in V \mid |w|_a > 0\}$ of symbols that appear in w.

2.2 Grammar Prerequisites

The rewriting systems are string-processing systems, evolving according to the rewriting rules, replacing a substring of the processed string by another string.

2.2.1 Grammars

A *grammar* is a device (with a finite description) *generating* a language.

A *Chomsky grammar* is a quadruple $G = (N, T, S, P)$, where N and T are disjoint alphabets, $S \in N$, $P \subset_{fin} V^*NV^* \times V^*$, where $V = N \cup T$. The elements of a grammar are called the *non-terminal alphabet*, the *terminal alphabet*, the *axiom* (or start symbol), and the set of *productions*, respectively. The rules $(u, v) \in P$ are written as $u \to v$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

For $x, y \in V^*$ we write $x \Rightarrow y$ (x *directly derives* y) if $\exists x', x'' \in V^*, \exists (u \to v) \in P : x = x'ux'', y = x'vx''$. The relation $\Rightarrow^*$ (derives in 0 or more steps) is defined as the reflexive transitive closure of $\Rightarrow$. $SF(G) = \{w \in V^* \mid S \Rightarrow^* w\}$ is the set of *sentential forms* generated by G. The language generated by G is denoted by $L(G) = SF(G) \cap T^*$. We denote the family of all finite languages by FIN.

The Chomsky grammars are classified in the following way:

- arbitrary(type 0): $|u|_N \geq 1$ for each $(u \to v) \in P$
- length-increasing (monotone): $|u| \leq |v|$ (and $|u|_N \geq 1$)
- context-sensitive (type 1): for each $(u \to v) \in P - \{S \to \lambda\}$ there exist $u', u'' \in V^*$, $x \in V^+$ and $A \in N$ such that $u = u'Au'', v = u'xu''$
  AND ALSO if $(S \to \lambda) \in P$, then $|v|_S = 0$ for all $(u \to v) \in P$
- context-free (type 2): $u \in N$ for all $(u \to v) \in P$
- linear: $u \in N$ and $|v|_N \leq 1$ for all $(u \to v) \in P$
- right-linear: $u \in N$ and $v \in T^* \cup T^*N$ for all $(u \to v) \in P$
- regular (type 3): $u \in N$ and $v \in T \cup TN$ for all $(u \to v) \in P - \{S \to \lambda\}$
  AND ALSO if $(S \to \lambda) \in P$, then $|v|_S = 0$ for all $u \to v$ in P

The families of languages, generated by the classes of grammars above, are RE, MON, CS, CF, LIN, $RLIN$, REG and their relationships are as follows (FIN stands for finite languages):

$$FIN \subsetneq REG = RLIN \subsetneq LIN \subsetneq CF \subsetneq CS = MON \subsetneq RE.$$

The *Dyck language* over $T_m = \bigcup_{1 \leq i \leq m} \{[_i\ ,\ ]_i\}$ is the context-free language generated by $G = (\{S\}, T_m, S, \{S \to \lambda, S \to SS\} \cup \{S \to [_i S]_i \mid 1 \leq i \leq m\})$, a language of all strings of nested parentheses of m kinds.

2.2.2 Programmed Grammars

Let us recall a class of regulated grammars - the programmed grammars. These grammars have been used to prove a number of results on the evolution–communication P systems.[1]

[1] So-called graph-controlled grammars have certain advantages over programmed grammars, but we introduce programmed grammars, in particular, for historical reasons.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Definition 2.2.1 *A* programmed grammar *(with appearance checking) is a system $G = (N, T, S, R)$, where N is a finite set of non-terminal symbols, T is a finite set of terminal symbols ($N \cap T = \emptyset$), $S \in N$ is the start symbol and R is a finite set of triples of the form*

$$(r : \alpha \to \beta, \sigma(r), \varphi(r)), \tag{2.1}$$

where r is a label of a rewriting rule $\alpha \to \beta$, $Lab(R) = \{r \mid (r : \alpha \to \beta, \sigma(r), \varphi(r))\}$ is the set of labels of rules in R and $\sigma, \varphi : Lab(R) \longrightarrow 2^{Lab(R)}$; $\sigma(r), \varphi(r)$ are called the success field and the failure field of r, respectively.

Definition 2.2.2 *The* language *generated by a programmed grammar.*
Let $(r : \alpha \to \beta, \sigma(r), \varphi(r)) \in R$. 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 α 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 labelled 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 2.2.1 *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 $w' = uAv$, $u, v \in T^*$, $A \in N$, $(r' : A \to y, \sigma(r'), \varphi(r')) \in R$, 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 R, and add r to the success and failure fields of all terminal rules without changing the language. An advantage of this fact is taken in the universality proofs.*

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).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Note 2.2.1 *From now on by programmed grammars we will assume programmed grammars with appearance checking with* context-free rules*, i.e.,* $|\alpha| = 1$ *in (2.1).*

Remark 2.2.2 *In the universality proofs simulating the programmed grammars with appearance checking, pairs* $\langle S = w_1, p_0 \rangle$, $\cdots$, $\langle w_m, p_{m-1} \rangle$, $\langle w_{m+1} = x, p_m \rangle$ *are considered for derivation* $(p_1, S = w_1) \Rightarrow \cdots \Rightarrow (p_m, w_m)$, $w_m \Rightarrow_{p_m} w_{m+1} = x$*. 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.*

Lemma 2.2.1 *The class of programmed grammars with appearance checking generates exactly the family of recursively enumerable languages.*

For the proof, see Theorem 1.2.5 in [78].

2.2.3 Matrix Grammars

In many universality proofs of P systems *matrix grammars with appearance checking* in the binary normal form are used. Such a grammar is a construct $G = (N, T, S, P, F)$, with $N = \{S, \#\} \cup N_1 \cup N_2$ (these three sets are mutually disjoint, the elements of N_1 are the "control non-terminals", the elements of N_2 are the "literal non-terminals"), and with the matrices from P in one of the following forms: $(S \to X_{init} A_{init})$ (unique), $(X \to Y, A \to x)$, $(X \to Y, A \to \#)$, $(X \to \lambda, A \to x)$, where $X_{init}, X, Y \in N_1$, $A_{init}, A \in N_2$ and $x \in (N_2 \cup T)^*, |x| \leq 2$. F is the set of the rules that can be used in appearance-checking mode (skipped only if not applicable), and this set consists of exactly all the rules producing #.

In matrix grammars *without appearance checking*, the rules of the third form are not allowed, and $F = \emptyset$. Matrix grammars with appearance checking generate RE, while the family of languages generated by matrix grammars without appearance checking is denoted by MAT.

2.2.4 Lindenmayer Systems

A 0L (interactionless Lindenmayer) system is a triple $G = (V, w, P)$, where V is an alphabet, $w \in V^*$ is an axiom and P is a finite set of rules, associating to every $a \in V$ at least one rule $a \to v, v \in V^*$. We can also represent

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

productions P as a mapping $p : V \to FIN - \{\emptyset\}$. For $x = a_1 \cdots a_n \in V^*, y \in V^*$ we write $x \Rightarrow y$ (x directly derives y) if $y \in p(a_1) \cdots p(a_n)$. The relation $\Rightarrow^*$ (derives in 0 or more steps) is defined as the reflexive transitive closure of $\Rightarrow$. The generated language is $L(G) = \{x \in V^* \mid w \Rightarrow^* x\}$

G is called *deterministic* if $|p(a)| = 1$ for all $a \in V$.

A *tabled* 0L system is such a system $G = (V, w, P_1, \cdots, P_n)$ that every (V, w, P_i) is a 0L system; the production sets P_i are called tables; $x \Rightarrow y$ if and only if $x = a_1 \cdots a_n \Rightarrow y$ and $y \in p_i(a_1) \cdots p_i(a_n)$ for some i. A $T0L$ system is called deterministic if $|p_i(a)| = 1$ for all $a \in V$ for every table P_i.

A Lindenmayer system is called *extended* if a terminal subset $T \subseteq V$ is distinguished and $L(G) = \{x \in T^* \mid w \Rightarrow x\}$. Then we write $G = (V, T, w, P_1, \cdots, P_n)$.

The families of ((E)xtended)((D)eterministic)((T)abled)$0L$ systems are denoted by $(E)(D)(T)0L$, respectively.

Lemma 2.2.2 *(Normal Form) For each $L \in ET0L$ there is an extended tabled Lindenmayer system $G = (V, T, P_1, P_2, w_0)$ with two tables generating L such that the terminals are only trivially rewritten, i.e., for any $a \in T$, if $(a \to \alpha) \in P_1 \cup P_2$ then $\alpha = a$.*

2.3 Automata and Machines Prerequisites

2.3.1 Finite Automata and Transducers. PDAs

Automata are analyzing machines, or recognizers. A (non-deterministic) finite automaton is a quintuple $A = (Q, T, s_0, F, \delta)$, where Q and T are disjoint alphabets, $s_0 \in Q$ and $F \subseteq Q$ are the initial state and the set of final states, respectively, and $\delta : Q \times T \to 2^Q$ is the transition function. The automaton is called *deterministic* if $|\delta(s, a) \leq 1|$ for all $s \in Q, a \in T$.

The transition function is extended to $\delta' : Q \times T^* \to 2^Q$ as follows: $\delta'(s, \lambda) = \{s\}$ for all $s \in Q$; $\delta'(s, aw) = \bigcup_{t \in \delta(s,a)} \delta'(t, w)$ for all $s \in Q, a \in T, w \in T^*$. The language, recognized by a finite automaton A is $L(A) = \{w \in T^* \mid \delta'(s_0, w) \in F\}$. The power of finite automata is not decreased if we restrict them to be deterministic. The power is not increased if we allow λ-transitions, i.e., if δ is defined on $Q \times (T \cup \{\lambda\})$, or when the input is scanned in a two-way manner, without changing its symbols.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

A finite transducer (a finite automaton with output) is a tuple $R = (Q, T, V, s_0, F, \delta)$, where Q, T, s_0, F are like in a finite automaton, V is the output alphabet and $\delta : Q \times (T \cup \{\lambda\}) \to 2^{Q \times (V \cup \{\lambda\})}$.

The transition function is then extended to $\delta' : Q \times T^* \to 2^{Q \times V^*}$. The output $M(x)$ of M on the input x is $\{y \in V^* \mid (s, y) \in \delta'(s_0, x) \cap F \times V^* \text{ for some } s\}$. The image of a language $L \subseteq T^*$ under a transduction is $L' = M(L) = \{M(x) \mid x \in L\}$.

A *pushdown* automaton is a finite automaton, equipped with a pushdown - a last-in-first-out type storage.

2.3.2 Turing Machines. LBAs

A Turing machine is a tuple $M = (Q, V, T, B, s_0, F, \delta)$, where Q, V are the set of states and the alphabet, $T \subseteq V$ is the input alphabet, $B \in V - T$ is the blank symbol, $s_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, $D = \{L, N, R\}$ and $\delta : (Q - F) \times V \to 2^{Q \times V \times D}$ is the transition function. If $(t, b, d) \in \delta(s, a)$ for $s, t \in Q$, $a, b \in V$ and $d \in D$, then the machine, being in state s, rewrites the symbol a into b, does not move the head if $d = N$, moves the head to the left if $d = L$ or to the right if $d = R$, and passes to state t. The machine M is called *deterministic* if $|\delta(s, a)| \leq 1$ for all $s \in Q - F$, $a \in V$.

A configuration of a Turing machine is a string vsw, where $s \in Q$ and $v \in V^* - BV^*$, $w \in V^* - V^*B$. A direct transition is defined as follows:

Let vsw be the current configuration, let $wB = aw'$ and $Bv = v'c$,[2] also let $(t, b, d) \in \delta(s, a)$. For $d = L$, $d = N$ and $d = R$, the resulting string then is $v'tcbw'$, $v'ctbw'$, and $v'cbtw'$, respectively. Finally, eliminate symbols B at the ends of the string to obtain the configuration representing the result of the transition. Again, $\Rightarrow^*$ (transition in 0 or more steps) is defined as the reflexive transitive closure of $\Rightarrow$.

The language recognized by a Turing machine M is $L(M) = \{w \in T^* \mid s_0 w \Rightarrow^* xsy \in V^* F V^*\}$. A Turing machine can be also viewed as a generative device, if it starts with the fixed configuration s_0, or a function computing device.

A *linearly bounded automaton* is a Turing machine restricted to the use of working space linearly bounded with respect to the length of the input.

[2] i.e., a is the first symbol of wB and w' is the rest of wB; c is the last symbol of v and v' is the rest of Bv.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Formally, there exist non-negative integers a and b such that for any configuration represented by xsy such that $s_0w \Rightarrow^* xsy$ we have $|xy| \leq a|w| + b$.

2.3.3 Counter Automata

A counter is as a memory storage able to hold (unbounded) non-negative integer values, that can be incremented, decremented or tested for zero (a counter is called empty if its value is zero). [3]

For the moment, let us consider the case of accepting vectors corresponding to the initial values of the input counters. A counter automaton is a tuple $M = (d, Q, q_0, q_f, P)$, where

- d is the number of counters (we will use the notation $D = \{1, \cdots, d\}$);
- Q is a finite set of states;
- $q_0 \in Q$ is the initial state;
- $q_f \in Q$ is the final state;
- P is a finite set of instructions.

The instructions are of the forms $(q_i \rightarrow q_l, k\gamma)$, with $q_i, q_j \in Q$, $q_i \neq q_f$, $k \in D$, $\gamma \in \{+, -, = 0\}$ (changing the state from q_i to q_l and applying operation γ to counter k). The operations are *increment* (add one to the value of the counter), *decrement* (subtract one from the value of the counter) and *zero-test* (test whether the value of the counter is zero or not), respectively. If an empty counter is decremented or a non-empty counter is tested for zero, then the computation is blocked. One can also speak about the *Stop* instruction of the counter automaton, assigned to the final state q_f.

A transition of the counter automaton consists in updating/checking the value of a counter according to an instruction of one of the types described above and by changing the current state to another one. The computation starts in state q_0 with all counters equal to zero, except the input ones. A

[3] From the grammar viewpoint, the notion of a counter (counter automaton) is very similar to that of a pushdown with one symbol (an automaton with multiple one-symbol pushdowns).

Alternatively, a counter (counter machine) is a blank tape with an end marker, infinite in one direction (a non-writing Turing machine with a finite, one way input tape and a number of counters).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

computation accepts a vector of initial values of the input counters if it halts in state q_f (without loss of generality we may assume that in this case the input counters are empty).

Again without loss of generality (unless a deterministic case is considered) we may assume that the input counters are the first m counters, that they are never incremented, and the zero-test is only done, and for all input counters, at the end of the computation.

To generalize the behavior of counter automata from accepting vectors to accepting words, the input counters are replaced by an input tape. Such devices (denoted by (d, T, q_0, q_f, P), where T is the alphabet of the input tape) additionally use instructions of the form $(q_i \to q_l, read(a))$, $a \in T$.

It is worth mentioning that counter automata can be used as generative devices, starting from empty counters, the result being the vector of values of the output counters when the automaton halts in state q_f (without loss of generality we may assume that other counters are empty).

Again, without loss of generality we may assume that the output counters are never decremented or zero-tested.

Generalizing also the generating from vectors to words is done in a similar way: replacing the output counters by an output tape, and let devices (d, T, q_0, q_f, P), where T is the alphabet of the output tape, additionally use instructions of the form $(q_i \to q_l, write(a))$.

One can consider both input and output, but we do not go here into details. The result we will use is that counter automata are computationally complete, and only two counters are needed (besides the counters or tape for input or output). This closely relates to the register machines, described in Subsection 2.3.6.

2.3.4 Conflicting Counters

A special variant of counter automata uses a set C of pairs $\{i, j\}$ with $i, j \in Q$ and $i \neq j$. As a part of the semantics of the *counter automaton with conflicting counters* $M = (d, Q, q_0, q_f, P, C)$, the automaton stops without yielding a result whenever it reaches a configuration where, for any pair of conflicting counters, both are non-empty.

Given an arbitrary counter automaton, we can easily construct an equivalent counter automaton with conflicting counters: For every counter i which shall also be tested for zero, we add a conflicting counter $\bar{\imath}$; then we replace all "test for zero" instructions $(l \to l', i = 0)$ by the sequence of instruc-

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

tions $(l \to l'', \bar{\imath}+)$, $(l'' \to l', \bar{\imath}-)$. Thus, in counter automata with conflicting counters we only use "increment" instructions and "decrement" instructions, whereas the "test for zero" instructions are replaced by the special conflicting counters semantics.

2.3.5 Partially Blind Counter Automata

Another special variant of counter automata called a *partially blind* counter automaton, see [102], is a restricted type of counter automata which has a finite number (say, m) of counters which can be incremented and decremented, but cannot be tested for zero. If there is an attempt to decrement a zero counter, the system aborts and does not accept.[4]

We will consider the case of accepting vectors (the first k counters are input counters). The device (also called a partially blind multicounter machine) starts with all counters except the input ones set to zero. The input tuple is accepted if the system reaches a halting state and all the counters are zero.

For notational convenience, we will denote the family of sets of tuples of natural numbers accepted by some PBCA as $aPBLIND$ and the family of sets of tuples of natural numbers accepted by PBCAs with m counters as $aPBLIND(m)$.

One can also consider a partially blind counter automaton as a generator of vectors of non-negative integers, see [121], by distinguishing output counters (such device is also called a partially blind multicounter generator).

We will denote the family of sets of tuples of natural numbers generated by some PBCA as $PBLIND$ and the family of sets of tuples of natural numbers generated by PBCAs with m counters as $PBLIND(m)$.

2.3.6 Register Machines

An n-register machine is a construct $M = (n, P, l_0, l_h)$ where:

- n is the number of registers;

[4]There is a strong similarity between partially blind counter automata and matrix grammars without appearance checking. In fact, $PsMAT = PBLIND = aPBLIND$, see also [84] and [120].

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- P is a set of labelled instructions of the form $l : (op(i), l', l'')$, where $op(i)$ is an operation on register i of M; symbols l, l', l'' belong to the set of labels associated in a one-to-one manner with instructions of P;

- l_0 is the initial label;

- l_h is the final label.

The instructions allowed by an n-register machine are:

- $l : (A(i), l', l'')$ – ADD instruction, add one to the contents of register i and proceed to instruction l' or to instruction l'' ($l = l''$ for the deterministic variant, the instruction can then be written as $l : (A(i), l')$);

- $l : (S(i), l', l'')$ – SUB instruction, jump to register l'' if the register i is null; otherwise subtract one from register i and jump to instruction labelled l' (these two cases are often called zero-test and decrement).

- l_h : *halt* – finish the computation. This is a unique instruction with label h.

If a register machine $M = (n, P, l_0, l_h)$, starting from the instruction labelled l_0 with all registers being empty, stops by halting with value n_j in every register j, $1 \leq j \leq k$ and the contents of registers $k+1, \cdots, n$ being empty, then it generates a vector $(n_1, \cdots, n_k) \in \mathbb{N}^k$. Any recursively enumerable set of vectors of non-negative integers can be generated by a register machine.

A register machine $M = (n, P, l_0, l_h)$ accepts a vector $(n_1, \cdots, n_k) \in \mathbb{N}^k$ if and only if, starting from the instruction labelled i, with register j having value n_j for $1 \leq j \leq k$, and the contents of registers $k+1, \cdots, n$ being empty, the machine stops by the *halt* instruction with all registers being empty.

Lemma 2.3.1 *For any partial recursive function $f : \mathbb{N}^\alpha \to \mathbb{N}^\beta$ there exists a deterministic $(\max\{\alpha, \beta\} + 2)$–register machine M computing f in such a way that, when starting with $(n_1, \cdots, n_\alpha) \in \mathbb{N}^\alpha$ in registers 1 to α, M has computed $f(n_1, \cdots n_\alpha) = (r_1, \cdots r_\beta)$ if it halts in the final label h with registers 1 to β containing r_1 to r_β (and with all other registers being empty); if the final label cannot be reached, $f(n_1, \cdots, n_\alpha)$ remains undefined.*

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

It is also known that, using $\alpha+\beta+2$ registers instead of $\max\{\alpha,\beta\}+2$, one can additionally require, without restricting the generality, that only ADD instructions are associated to the β output registers, only SUB instructions are associated to the α input registers, and the $\alpha+2$ registers (input ones and the working ones), are empty at the end of the computation.

As a corollary of this lemma, deterministic $(\alpha+2)$-register machines can accept all recursively enumerable sets of α-dimensional vectors of non-negative integers, while non-deterministic $(\beta+2)$-register machines can generate all recursively enumerable sets of β-dimensional vectors of non-negative integers.

One can generalize the output of register machines from vectors to words by replacing the output registers by an output tape. To work with this tape, the writing instructions $(l : write(a), l', l'')$ may be used. Register machines can generate/accept all recursively enumerable languages, and two registers in addition to an output/input tape are sufficient. We skip the aspect of accepting languages and related determinism issues.

You can see that the definitions of register machines and counter automata are quite similar, the formalisms are equivalent. However, it is sometimes more convenient to use one notion or the other one, depending on the following considerations:

Determinism. The register machines only make the "choice" during ADD instruction, and this is a choice between two alternatives (there is determinism if the alternatives are the same). This is more complicated for the counter automata. They make choice between different instructions starting from the same state.[5] *Determinism is more clear for the register machines* (accepting or computing functions).

Choices. To simulate the register machines, the choice of the next instruction (ADD case) should be explicitly implemented, while for the counter automata this corresponds to a simple selection of the instruction. *Choices are more automatic for the counter automata.*

Halting. The register machines halt when no operation is applicable. This is only possible, by definition, in the final state. The counter machines halt in the final state. One can either eliminate the cases when the counter machine inevitably blocks by constructing a *complete* automaton (using a

[5]There is determinism, e.g., if there are at most two instructions assigned to every state, and moreover, if they are two, then they must be a decrement and a zero-test of the same counter. In general, it is undecidable whether a counter automaton is deterministic or not.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

non-final sink state), or to check, when no operation is applicable, whether the state is actually final. *Halting is easier to check for the register machines.*

Cases. A simulation of register machines consists of two cases, and ADD is usually[6] much easier than SUB because it is non-cooperative and because its two subcases are equivalent. A simulation of counter automata typically consists of three cases, increment is usually simpler than decrement, which, in its turn, is usually simpler than zero-test. *It is easier to analyze the counter automata constructions*, because the complicated SUB instruction is syntactically split in decrement and zero-test.

[6]But not always, see, e.g., Theorem 5.6.2.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 3

Introduction to P Systems

3.1 P Systems Prerequisites

Membrane systems (also called P systems) were originally introduced by Gheorghe Păun in 1998 as a framework of formal computational models inspired by the structure and the functioning of living cells.

3.1.1 Basic Definitions

A membrane is a separator of the regions, here - the inside and the outside. The regions contain objects and may allow different reactions to happen. Membranes permit certain objects to pass through, and also serve as a communication channels between the regions. These notions outline the framework of P systems.

The aim of P systems is not to model the processes of biological cells, but rather to study the computational aspect of different features of membranes. To do this, only a few of basic principles are considered in a rather abstract way.[1]

A *membrane structure* is a system of membranes, say $\{l_i \mid 1 \leq i \leq m\}$ with $\prec$: a transitive ($l_i \prec l_j, l_j \prec l_k \Rightarrow l_i \prec l_k$) and anti-reflexive($\neg l_i \prec l_i$) (strict) partial order relation "inside". Moreover, the *skin membrane* l_s must exist: $l_i \prec l_s$, $1 \leq i \leq m$, $i \neq s$.

[1] A number of articles on modelling biological (and not only biological) processes have recently appeared, see. e.g., [71]. However, we will not speak about this direction in the present work.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

An *elementary* membrane is a membrane l_j: $\neg l_i \prec l_j$, $1 \leq i \leq m$. The child membrane of a (non-elementary) membrane l_k is any membrane $l_i \prec l_j$, such that no membrane l_j is between them ($\neg \exists l_j$: $l_i \prec l_j \prec l_k$). Membranes delimit regions: we will call the environment r_0 the outside region of the skin membrane. Every membrane l_i defines a region r_i inside it and outside its children membranes.

The membrane structure can be represented by an Euler-Venn diagram, or by a (directed, with unordered descendants) rooted tree, where the nodes are the regions: the root is the environment, its only descendant is the skin region; the edges are the membranes: the in-edge of a node is a membrane surrounding that region, and the out-edges of a node are the children membranes (typically, the structure is simplified by taking the skin as the root). The membrane structure can also be represented by a string of matching parenthesis (looking like one-dimensional Euler-Venn diagram), which would be any string $w \in D_m$ (the Dyck language with m kinds of brackets), such that $w \notin D_m D_m$ (the skin membrane exists) and $|w|_a = 1$ for every $a \in T_m = \bigcup_{1 \leq i \leq m} \{[_i \ , \ ]_i\}$ (as you will see later, for P systems with dynamic membrane structure the last condition is dropped). A string $\cdots [_i \ \cdots [_j \ \cdots \ ]_j \cdots \ ]_i \cdots$ represents the structure where membrane l_j is inside membrane l_i.

3.1.2 Transitional P Systems

Each region contains a multiset of *objects*, represented by symbols from the alphabet, say O. Each region r_i contains a set of (evolution) rules of the form $u \to v$, $u \in O^+$, $v \in (O \times Tar)^*$, where $Tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}\}$. The string u represents a multiset of objects, consumed by the reaction, and the string v represents a multiset of objects produced, together with their *target indications*, written as subscripts (target indication *here* is typically omitted). The objects with subscript *out* are sent out through membrane l_i and the objects with subscript in_j have to immediately be sent inside membrane l_j. For a rule having targets in_j to be applicable, l_j must be a child membrane of l_i.

The transition from a *configuration* of a P system to the next one consists in a *non-deterministic* and *maximally parallel* application of the rules in all the regions. This means that the rules are assigned to objects in a non-deterministic way and no rule is applicable to the remaining objects (the objects are exhausted by the rules). Thus, "some maximal" multiset of rules

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

from each region is chosen to be applied, the assigned multiset of objects is removed from the system and the resulting multisets of objects are added to the system and the objects with targets are moved to the corresponding regions.

A configuration of a P system is called *halting*, if no rules are applicable in any region. A *computation* is a sequence of transitions; it is called halting if it reaches a halting configuration.

3.1.3 Basic Variants and Additional Features

So far, a general definition was given, allowing evolution rules with full cooperation. An evolution rule $u \to v$ is called non-cooperative if $|u| = 1$.

- P systems with *catalysts*. In this variant, a subset of objects $C \subseteq O$ is distinguished, called catalysts. The only rules allowed are of type $a \to v$, $a \in O - C$, $v \in (O - C)^*$ (non-cooperative, without catalysts) and of type $ca \to cv$, $c \in C$, $a \in O - C$, $v \in (O - C)^*$ (catalytic). The catalysts are neither created nor destroyed, they neither evolve nor go through membranes, and they only participate in the catalytic rules.

- P systems with *m-stable* catalysts. The evolution rules containing catalysts are in the form $c_i a \to c_j v$ where $c \in C$ and $1 \leq i, j \leq m$, where i, j are states associated with c. Sometimes one defines an m-stable catalysts as a set $\{c_i \mid 1 \leq i \leq m\}$ of all its states.

- P systems with *mobile* catalysts. This is an extension of the catalytic variant by permitting catalysts to move between regions. In this case the catalytic rules are of the form $ca \to c_{tar} v$ where $tar \in \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$.

- P systems with *i/o* communication. The target indications are now $\{here, out, in\}$, the indication in meaning "into any inner membrane".

- The (strong) *priority* relation among the rules. A partial order relation is defined on the set of rules of each region. A rule is only applicable if no rule of a higher priority is applied in the same step.

- The *weak* priority relation among the rules. A partial order relation is defined on the set of rules of each region. A rule is only applicable if no more rules of a higher priority could be applied instead.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- P systems with promoters/inhibitors. In the case of promoters, the rules (reactions) are possible only in the presence of certain symbols which can evolve simultaneously with symbols whose evolution they support. On the contrary, inhibitors forbid certain rules (reactions). We denote these features by considering the rules of the form: $u \to v|_a$ or $u \to v|_{\neg a}$.

- Controlling membrane *permeability*. The membrane can be *dissolved*: the objects of a region corresponding to a dissolved membrane remain in the region surrounding it (corresponding to the parent membrane). Formally this is represented as $u \to v\delta$: after the rule evolving u into v is applied, the membrane is dissolved. The skin is never dissolved. The membrane can be made impermeable (no objects can pass through it) by a rule $u \to v\tau$ (details are skipped, see [163]).

The computational power of transitional P systems is studied with respect to number of membranes and the features used.

3.1.4 Formal Description

From now on, instead of "membrane l_i" and "region r_i", we will simply use the index. Also, when speaking of objects in membrane i, objects in region i are assumed.

We now describe a P system by a construct $\Pi = (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m)$, where O is the alphabet - the set of objects, μ is the (expression describing the) membrane structure, consisting of m membranes, w_i are the strings representing multisets of objects in regions $i : 1 \leq i \leq m$, R_i are the sets of evolution rules in regions $i : 1 \leq i \leq m$.

In a system with a static membrane structure (μ is fixed) the *configuration* can be described by a vector $C' = (w'_i)_{0 \leq i \leq m}$ of (strings describing the) multisets of objects in all the regions ($i = 0$ corresponds to the environment). Then the halting computation is just a finite sequence of vectors of multisets. If the membrane structure is dynamic, then its description must also be included in the configuration.

3.1.5 Purely Communicative Systems

The computation can be made only by moving objects between the regions. The most studied model of such P systems is called P systems with symport /

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

antiport rules. The *communicative* rules are now assigned to membranes and govern the object transport through them, as opposed to evolution rules, assigned to regions, describing the reactions of objects.

There are two types of rules considered: *symport* rules of types (x, out) and (x, in), $x, y \in O^+$ (meaning that the multiset x simultaneously exits the membrane or that the multiset x simultaneously enters the membrane) and *antiport* rules of type $(y, out; z, in)$, $y, x \in O^+$ (y exits and z enters the membrane at the same time).

It is necessary to mention that the work of these systems is defined to take place in the infinite environment (a subset $E \subseteq O$ is specified) and the environment (initially empty in the definition of transitional P systems) contains an infinite supply of objects from E. The computational power of the purely communicative P systems is studied with respect to the number m of membranes and the bounds p, q on the weights of symport / antiport rules ($|x| \leq p$, $max(|y|, |z|) \leq q$ for all symport and antiport rules), as well as the size of the alphabet.

A formal definition is given in Chapter 5.

3.1.6 Evolution–Communication P Systems

This variant combines symport / antiport rules associated to membranes with evolution rules associated to the regions. However, the evolution rules are typically non-cooperative and without targets, and the environment is typically empty in the beginning of the computation.

One of the extensions considered is to allow target indications in the evolution rules; the other one is to allow the environment to initially contain an infinite supply of objects (for example, of one kind). One of the restricted variants considered is proton pumping P systems: a subset of objects called protons is specified, they are neither created nor destroyed by evolution rules, while some proton participates in every symport / antiport rule. By analogy, a proton can be called a "catalyst of communication".

A formal definition is given in Chapter 4.

3.1.7 Tissue P Systems

In so-called tissue P systems, the membranes are replaced by channels, connecting regions (except a unique region called the environment, regions are called cells) in a graph, not necessarily a tree.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

For instance, a meaning of a symport / antiport rule x/y associated to channel (i, j) is that the multiset represented by x is moved from region i to region j, while the multiset represented by y is moved from region j to region i. ($x, y \in O^+$ in the antiport case; one of them is λ in the symport case.)

Moreover, it is typically assumed that the rules associated to the same channel are not applied in parallel.

A formal definition is given in Chapter 5.

3.1.8 Further Models

Many variants have been introduced in the literature. Below a short summary of some of them is given.

By structure of objects: instead of symbol objects from a finite set, one can consider objects with structure (strings, arrays, etc.) with various rules (rewriting, rewriting with replication, splicing, etc.).

By membrane operations: certain additional operations can be used, like membrane creation, membrane division, etc. In so-called active membranes a polarization is associated to each membrane.

By limited parallelism: in tissue P systems, at most one rule per channel can be applied at any computation step; in active membranes at most one rule per membrane (except evolution rules without targets) can be applied at any computation step; some variants (like sequential P systems, k-sequential P systems, P systems with minimal parallelism, etc.) do not assume maximally parallel use of (some) rules.

By additional control: variants of P systems have been introduced with various regulations of the computations, e.g. energy.

Since the scope of this thesis is restricted to P systems with symport / antiport of objects, none of the above will be discussed here in detail, except division of an elementary membrane $[_h\ a\ ]_h \to [_h\ b\ ]_h[_h\ c\ ]_h$, where $h \in H$ (the set of membrane labels, no longer uniquely identifying membranes), and $a, b \in O$.

3.2 Computing with P Systems

Finally we discuss why the application of rules of a P system is called computation and what can be understood by its result.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

3.2.1 Generating

The output region i_0 is specified in the definition of a P system. The *output* alphabet $T \subseteq O$ can be specified, and then the objects of the result from $O - T$ are ignored. (If T is omitted in the definition of a P system, it can be typically assumed to be the set of all objects, except those that can never appear in the output region in a halting configuration).

Languages (external output): the *result* of a (halting) computation is a string of objects sent to the environment (the order of the objects, sent out of the system at the same time, is arbitrary). The language generated by a P system Π (by means of all halting computations) is denoted by $L(\Pi)$.

Vector sets or number sets: multiplicities of objects or their total number are counted in a designated region (can also be the environment). Then it is said that a P system Π generates a vector set $Ps(\Pi)$ or a number set $N(\Pi)$, respectively. In principle, one can speak about more than one output region (distributed output).

3.2.2 Accepting

The input region i_0 is specified in the definition of a P system. The *input* alphabet $\Sigma \subseteq O$ is typically specified.

Accepting P systems: before it starts, a P system receives a vector of input objects in the input region. A vector/number is accepted if the system eventually halts. It is then said that a P system Π accepts a vector set $Ps_a(\Pi)$ or a number set $N_a(\Pi)$, respectively.

Distributed input: in some variants, the input is placed in more than one region of the P system. This is especially useful if the input needs to be represented by just one symbol.

P automata (initial mode): the input word is given to the P system in the environment, symbol by symbol (i-th symbol can be used in step i). A word is accepted by a P system if it uses each input symbol immediately after it is available, and eventually halts. The language accepted by a P automaton Π is denoted by $A_I(\Pi)$.

3.2.3 Processing

One can consider P systems with both input and output.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

3.2.4 Deciding

This is a particular case of processing. Typically one requires that the system halts on every input, ejecting the answer, i.e., either one copy of a special object `yes`, or one copy of a special object `no` into the environment. Usually one also requires that the answer is sent into the environment in the last step of the computation.

3.3 Properties

3.3.1 Decidability, Completeness, Universality

A reader interested in a more detailed discussion of this topic is referred to, e.g., [79].

Consider a family F of computing devices. A property P of devices can be defined as a subset of F (we say that a device $D \in F$ satisfies property P if and only if $P \in D$).

A property P is called *decidable* if and only if its characteristic function is computable (and it is called undecidable otherwise). When we say that a computing device is given, it is understood that its description is given.

Probably, the most interesting property of computing devices (assuming that the possible input is included in the description) is *halting*. Let us restrict the discussion to the devices producing result if and only they halt (halting is the same as producing result).

In case of generative devices, this corresponds to non-emptiness. In case of accepting devices, non-emptiness corresponds to halting on *some* input. In case of processing devices, it is often desired to halt on *every* input (this relates to computing total functions or accepting recursive sets).

We will say that a family of devices that generate/accept a set of numbers (vectors, words) is *computationally complete* if it generates NRE ($PsRE$, RE). The halting problem is undecidable for any computationally complete family of devices (the devices are said to be unpredictable).

A computational device D with input (there are also definitions for generative devices, but we will not go into more details here) is called *universal* (with respect to a family F' of computational devices) if D can simulate the behavior of any device $D' \in F'$, given as input (in some encoding). Slightly more formally, this implies that there exist a ("simple") encoding function e and a ("simple") decoding function d such that the decoding d of the result

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

of D, given as input the encoding e of a description of D' will be the same as the result of D'.

Sometimes, the family F' is not specified: we say that D is universal (or devices of $F \ni D$ are universal), understanding that D is universal with respect to some computationally complete family of devices.

It is generally believed that computational completeness implies universality; it is often said that *some class of P systems is universal* when, in fact, its computational completeness has been proved. Conversely, most of universality proofs for P systems are by establishing the computational completeness of the corresponding class.

3.3.2 Determinism and Confluence

For a rewriting system RS, we write $C \Rightarrow C'$ if the system allows a direct transition from an instantaneous description C to an instantaneous description C' (C' is then called a next instantaneous description of C). The relation $\Rightarrow^*$ is a reflexive and transitive closure of $\Rightarrow$. For any rewriting system, we use the word *configuration* to mean any instantaneous description reachable from the starting one.

Definition 3.3.1 *A configuration of a rewriting system is called* halting *if no rules of the system can be applied to it.*

In this work we will only talk about rewriting systems, producing the result at halting.

Definition 3.3.2 *A rewriting system is called* deterministic *if for every accessible non-halting configuration C the next configuration is unique.*

Definition 3.3.3 *A rewriting system is called* confluent *if, for every starting configuration C_0 one of the following holds: either all the computations are non-halting, or there exists a configuration C_h, such that all the computations halt in C_h.*

Notice that, if starting at some configuration C all the computations halt, then there exists $m \geq 0$, such that all the computations starting from C halt in at most m steps.

We will now introduce a weaker definition of the confluence of systems, with the computations "unavoidably leading" to the same result, but not necessarily in a bounded number of steps.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Definition 3.3.4 *A rewriting system is* ultimately confluent *if either a) no computations halt or b) there exists such a halting configuration C_h, that for any configuration C we have $C \Rightarrow^* C_h$.*

This property implies two facts: if a halting computation exists, then

1. the halting configuration is unique (C_h),
2. C_h is reachable from any configuration.

From now on we will only consider rewriting systems producing the result at halting. Let us consider the graph of all reachable configurations (the arc from configuration C to configuration C' means that C can derive C' in one step). The graph may be infinite. A node is called *final* if it has the out-degree 0.

A system is *deterministic* if all nodes have the out-degree of at most one (hence, there is at most one final node). A system is *confluent* if either there are no final nodes, or the final node is unique, and in that case the graph is finite and does not contain cycles. An *ultimately confluent* system may contain cycles, but either all nodes are non-final, or there is a final node reachable from any configuration.

Example: Consider a rewriting system with the initial configuration S and the rewriting rules:

$S \to SA$, $A \to \lambda$, $S \to a$.

Note that the system is not deterministic, and one can choose to apply the first rule an unbounded number of times, but from any configuration it is possible to arrive to the halting one $C_h = a$ by erasing all symbols A by means of the second rule (this is equally true no matter if the system is sequential, concurrent, or maximally parallel).

3.3.3 Uniform, Semi-uniform Families

A reader interested in a more detailed discussion of this topic is referred to, e.g., [152] and [176].

Consider some problem (let us formalize it as computing a function) $P : X \to Y$. For instance, $Y = \{\texttt{yes}, \texttt{no}\}$ when one speaks about a decisional problem. We will denote particular inputs by $A(n) \in X$, and their size by n.

A family F of P systems is called *uniform* (*semi-uniform*) if there is a function $u : \mathbb{N} \to F$ ($u : X \to F$), computable in polynomial time, that

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

maps numbers (problem inputs) to P systems (it is then understood that the family F is given together with the function u).

A problem P is said to be solvable by a semi-uniform family F if $u(A(n))$ halts and produces the result, corresponding to $P(A(n))$, for any input $A(n)$.

A problem P is said to be solvable by a uniform family F if, taking as input the (polynomially computable) encoding of $A(n)$, P system $u(n)$ halts and produces the result, corresponding to $P(A(n))$, for any input $A(n)$.

For decisional problems and P systems with external output, one requires that the corresponding P system sends one object, `yes` or `no` in the environment (sometimes one also requires the result is sent in the last step of the computation). In the general case, $P(A(n))$ should be obtained by applying a decoding function (computable in polynomial time) to the result of the computation of the corresponding P system.

The difference between uniform and semi-uniform solutions is that in the first case the problem input is encoded only in objects, while in the latter case it can be encoded in objects, membranes and rules.

Finally, one speaks about solving decisional problems by P systems in polynomial (or linear, etc.) time (i.e., number of steps). It is an open problem for all classes of P systems to constructively prove (or to disprove) that all decision problems solvable in polynomial time by a semi-uniform families of P systems are also solvable in polynomial time by a uniform family of P systems.

All time complexity measures in this subsection are understood as with respect to the size n of the problem input.

3.4 Maximal Parallelism. Simulator

The original aim of the research reported in this section was to produce a software tool to let a researcher (not necessarily in membrane computing) observe the evolution of maximally parallel multiset-rewriting systems of a general form, with permitting and forbidding contexts, browsing the configuration space by following transitions like following hyperlinks in the World-Wide Web.

The relationships of maximally parallel multiset-rewriting systems with other systems are investigated, such as Petri nets, different kinds of P systems, Lindenmayer systems, grammar systems, regulated grammars. The notion of maximal parallelism is expressed using linear programming – a

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

branch of optimization theory. The last sentence is the main reason to include this research in the thesis.

3.4.1 Introduction

There are numerous examples of simulators for deterministic and non-deterministic rewriting systems that are sequential or parallel. In this section we will focus on the non-deterministic parallel ones. Recall that in the process of the evolution, the system non-deterministically branches (chooses the next configuration among the possible ones).

The need for writing yet another simulator appeared because most of the simulators of *P systems with symbol objects* (i.e., of maximally parallel distributive multiset-rewriting systems) either simulate just the deterministic/confluent systems or resolve the non-determinism in a random way, see [38, 41, 43, 69, 74, 75, 199]. Clearly, for checking the theorems proved by construction of a system (by testing the examples), it is desired that either all branches of the needed number of steps of the evolution are explored, or one branch is selected by the user from the list of all possible branches at each step. Since the size of the solutions for the first approach may be too big to be studied without further tools, we focus on the latter one.

Thus, the main problem is to compute the set of all possible transitions given the rewriting system with the current configuration. The meaning of the word "browsing" in the title is analogous to its meaning in the World-Wide Web: following the transitions (hyperlinks) between the configurations (pages, documents), optionally remembering the path (history) to be able to come back.

As it was mentioned earlier, we need a method to compute the set of all possible transitions (configurations after 1 step), as a kernel for the configuration browser.

The systems we are considering are the maximally parallel multiset-rewriting systems with promoters/inhibitors. They can be viewed as the *Petri nets* with inhibitor arcs, with two differences: promoter arcs are added and the parallelism is maximal, in a way similar to the one described in Subsection 3.1.2. Notice that the maximality of parallelism can be avoided by adding rules $a \to a$ for all $a \in V$.

The maximally parallel multiset-rewriting can also be viewed as non-distributive variant of cooperative P systems with promoters/inhibitors. Encoding "being in a region i" in object a as, e.g., a_i, one obtains the (non-

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

distributive) *transitional P systems*. Using the same idea, *P systems with symport / antiport* are reduced to the cooperative multiset-rewriting systems.[2] The behavior of *P systems with active membranes* can be simulated by encoding the polarization of each membrane in one object, which acts as a 3-stable catalyst for the communication rules and a promoter for the evolution rules.

If we restrict the rules to be non-cooperative (context-free, $|u| = 1$), then we obtain a generalization of $0L$ systems; one can easily use promoters to simulate the behaviour of $ET0L$ systems, modulo the order of the symbols. A similar reasoning would establish a link between our systems and *grammar systems*, except in this case the control symbol is a (multi-stable) catalyst, not a promoter. The same is true for simulating the behaviour of sequential systems, for example of some *grammars with regulated rewriting*. One can find links with other rewriting systems.

The software can be used as a simulator for the systems, a debugger for the theorems proved in a constructive way, a browser of the configurations, a tool for the researcher, or a toy for a student. It was used as an engine of the simulator of P systems with symport / antiport by Vladimir Rogozhin, to check the theorems in [138] and [24]. It was also used for checking some constructions in [16].

3.4.2 Maximally Parallel Multiset Rewriting

Without restricting the generality, we can assume that the alphabet is ordered: $V = \{a_1, \cdots, a_k\}$.

A *multiset-rewriting system* is defined as a tuple $G = (V, R, w)$, where V is the alphabet, R is a finite set of rules of the form $r : u \to v$, where $u \in V^+$, $v \in V^*$, r is called the label of the rule and $w \in V^*$ is the initial configuration. The label uniquely defines the rule and the set of all labels is denoted by $Lab(R)$. A *multiset of rules* from R can be represented by a word over $Lab(R)$. Without restricting the generality, we can assume that the rule set is ordered: $R = \{r_1 : u_1 \to v_1, \cdots, r_m : u_m \to v_m\}$.

[2] It deserves some attention that the objects E present in the environment in infinite multiplicities (let us denote the set of all objects by O) are not represented in configurations (that are finite), except their copies that are inside the system. Instead, a rule $(ux, out; vy, in)$ assigned to the skin membrane (with label 1), where $u, v \in E^*$ and $x, y \in (O - E)^*$, is converted to a cooperative rewriting rule $h_1(ux)h_0(y) \to h_0(x)h_1(vy)$, where h_i are region-encoding morphisms: $h_i(a) = a_i$, $a \in O$ for $0 \leq i \leq 1$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

We now proceed to defining the parallel evolution step. It is said that $\rho = \prod_{j=1}^{m} {r_j}^{m(j)} \in Lab(R)^*$ is applicable to the configuration $w = \prod_{i=1}^{k} ({a_i}^{M(a_i)})$ if $\sum_{j=1}^{m} |u_j|_{a_i} * m(j) \leq M(a_i)$ for every $a_i \in V$, i.e., there are enough objects in the configuration to perform all the rules in ρ with a corresponding multiplicity. The result of applying ρ on w is

$$w' = \delta(w, \rho) = \prod_{i=1}^{k} ({a_i}^{M(a_i)+\sum_{j=1}^{m}(|v_j|_{a_i}-|u_j|_{a_i})m(j)}).$$

In words, w' was obtained from w by removing u_j for $m(j)$ times for all rules r_j, and then inserting v_j for $m(j)$ times for all rules r_j. In other words, symbols from u_j were independently replaced by those from v_j, multiple times.

Notice that w' consists of the symbols that did not react and of the symbols from the right-hand sides of the rules from R. The multiset of rules represented by ρ is *maximal* if it is applicable, and no rule $r_k \in R$ can be applied to the remaining symbols in the same step, i.e., for any $r_k \in R$ there is some

$$a_i \in V \text{ with } M(a_i) - \sum_{j=1}^{m} |u_j|_{a_i} m(j) < |u_k|_{a_i}.$$

The evolution of the system is non-deterministic: it evolves (in one step) from w by any maximal applicable multiset of rules to the corresponding configuration w', denoted by $w \Rightarrow^{\rho} w'$. The superscript ρ may be omitted, and $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$. For the rewriting system $G = (V, R, w)$, we define the set of sentential forms $SF(G)$ as $\{x \in V^* \mid w \Rightarrow^* x\}$, and we denote by $SF_n(G)$ the set of configurations that can be obtained from w in exactly n steps.

Example 1. Let us consider the grammar $G = (\{S, A, B, C\},\ \{p : \mathbf{SA} \to SAB, q : \mathbf{AB} \to ABC\},\ SAA)$, as well as the following derivations with respect to it:

$$\begin{aligned}
&\mathbf{SA}A \Rightarrow^{p} \mathbf{SAAB} \Rightarrow^{pq} SAABBC\\
&\mathbf{SAAB}BC \Rightarrow^{pq} \mathbf{SAAB}BBCC \Rightarrow^{pq} SAABBBBCCC \Rightarrow^{\cdots}\\
&\mathbf{SAAB}BC \Rightarrow^{pq} S\mathbf{AABB}BCC \Rightarrow^{qq} SAABBBCCCC \Rightarrow^{\cdots}\\
&S\mathbf{AABB}C \Rightarrow^{qq} \mathbf{SAAB}BCCC \Rightarrow^{pq} SAABBBCCCC \Rightarrow^{\cdots}\\
&S\mathbf{AABB}C \Rightarrow^{qq} S\mathbf{AABB}CCC \Rightarrow^{qq} SAABBCCCCC \Rightarrow^{\cdots}.
\end{aligned}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Hence, $SF_0(G) = \{SAA\}$, $SF_1(G) = \{SAAB\}$, $SF_{n+2}(G) = \{SA^2B^{2+n-k} C^{1+n+k} \mid 0 \leq k \leq n\}$, and at any step except the first two either both p and q are applied once, or q is applied twice.

Finally, to increase the power of these systems, let us add promoters and inhibitors to the system, see [52] (called in the regulated rewriting theory permitting and forbidding contexts, respectively, see [78]). The general form of the rules is extended to $r : u \to v|_{p,\neg q}$. The behavior of the system is defined as in the usual case, except for a given configuration w, instead of the whole set R of rules only the set of "active" (promoted and not inhibited) rules is considered. The rule r is promoted if $|w|_a \geq |p|_a$ for all $a \in V$. The rule r is inhibited if $|w|_a \geq |q|_a$ for all $a \in alph(q)$. A rule p without the inhibitor ($q = \lambda$) is never inhibited. A rule without the promoter ($p = \lambda$) is promoted by the definition.

3.4.3 Vector Representation. Simplex

In the simulator of the maximally parallel multiset-rewriting systems, the multisets were represented by vectors. A multiset M over $V = \{a_1, \cdots, a_k\}$ can be represented by a vector $(M(a_1), \cdots, M(a_k)) \in \mathbb{N}^k$.

Example 2. The rewriting system

$$G = (\{S, A, B, C\},\ \{p : SA \to SAB, q : AB \to ABC\},\ SAABBC)$$

with $SAABBC \Rightarrow^{pq} SAABBBCC$, $SAABBC \Rightarrow^{qq} SAABBCCC$ can be written in space $\mathbb{N}^4$ as

$$V = (4, R, (1, 2, 2, 1)),$$
$$R = \{p : (1, 1, 0, 0) \to (1, 1, 1, 0),\ q : (0, 1, 1, 0) \to (0, 1, 1, 1)\},$$

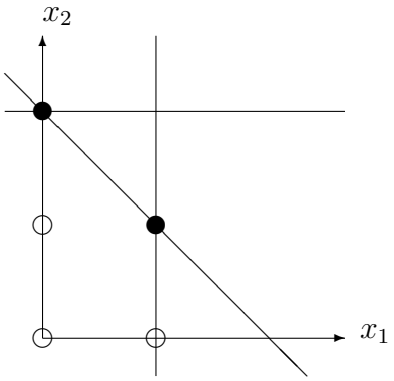
with $(1, 2, 2, 1) \Rightarrow^{pq} (1, 2, 3, 2)$, $(1, 2, 2, 1) \Rightarrow^{qq} (1, 2, 2, 3)$.

A *simplex* is a part of a finitely dimensional space, which is a set of solutions of a system of linear equations and inequalities, i.e., any intersection of a finite number of hyperplanes and semi-spaces.

All *multisets* of applicable rules form a simplex.

Example 3. For applying $p^{x_1}q^{x_2}$ to a in $p : (1, 1, 0, 0) \to b$, $q : (0, 1, 1, 0) \to c$, $a = (1, 2, 2, 1)$, where $b, c \in \mathbb{N}^4$, we obtain the following conditions ($x_1 \geq$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007



$$\begin{cases} x_1 & \leq & 1 \\ x_1 + x_2 & \leq & 2 \\ x_2 & \leq & 2 \\ 0 & \leq & 1 \\ x_1 & \rightarrow & max \\ x_2 & \rightarrow & max \end{cases}$$

Figure 3.1: Multi-criterial problem

$0,\ x_2 \geq 0$ are assumed):

$$\begin{cases} x_1 & \leq & 1 \\ x_1 + x_2 & \leq & 2 \\ x_2 & \leq & 2 \\ 0 & \leq & 1 \end{cases}$$

3.4.4 Solutions. Maximality. Optimization

The resulting strategies are the *integer* efficient (Pareto–optimal) solutions of the multi-criterial problem in Figure 3.1.

We now present a *recursive approach*. We start with the first rule ($r = 1$)

- Calculate the *maximal* possible applicable multiplicity x_r (1 in the example).
- For *all* numbers from x_r downto 0
  - calculate the *remaining* objects (in the example for $x_1 = 1$, $(0, 1, 2, 1)$ remains);
  - proceed with the *next* rule (if $r < m$);
  - otherwise, check the *maximality* (no more rules applicable to the remaining objects). If so, then *display*.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Because all space of applicable solutions is searched, the search should be done in a way as efficient as possible. Here is one idea in this respect: Divide and Conquer – *Independent* Symbols. The steps to follow are the next ones:

- select *applicable* rules
  (having enough objects, promoted and not inhibited);
- consider the *dependency graph*, whose nodes are the symbols, with two nodes connected if the corresponding symbols are in the *left-hand side* of the same applicable rule;
- consider the *components* of connectedness
  (maximal connected subgraphs). They correspond to the *independent* parts of the problem;
- solve each subproblem *separately*: the solution vector set is the *direct sum* of the vector sets of the subproblems.

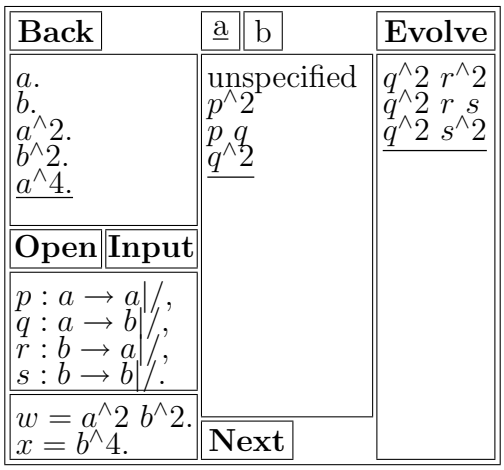


Figure 3.2: Simulator. The main interface

Figure 3.2 shows what the program interface looks like. It displays the current configuration w and the next configuration x in the bottom-right corner. Above it, the system rules are displayed. The button **Open** allows to load another system from a file. The button **Input** allows to manually enter the current configuration. Above these buttons, the history (list of

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

previous configurations) is displayed, and the last item is now selected. The button **Back** allows to return to the selected configuration.

In the middle-top part one can see the tabs corresponding to the independent sub-systems, and the first one is now selected. In the center, the list of choices of evolution is shown, for the symbols corresponding to the selected tab. The button **Next** allows to go to the next tab. The list of the maximal multisets of rules applicable to w is shown in the right part. Notice that making a choice for some sub-system acts as a filter for that list. Once some multiset is selected in the list on the right, the simulator re-computes x. The button **Evolve** allows to add w to the history, make x the current configuration and re-compute the independent sub-problems and the maximal multisets of applicable rules.

3.4.5 Summary

In this section we described ideas behind the construction of a simulator for non-deterministic parallel rewriting systems, namely, for maximally parallel multiset-rewriting systems with context.

The applications of such a simulator are mentioned, the systems simulated are related to other parallel rewriting systems, such as Petri nets, transitional P systems, P systems with symport / antiport, P systems with active membranes, Lindenmayer systems, grammar systems, regulated grammars.

An optimization of the algorithm of computing the set of possible transitions has been done before, for certain parallel rewriting systems with restricted forms of cooperation. It remains an open question how to compute the set of maximal multisets of rules applicable for a given configuration in a more efficient way in the general case. An interesting question would be to study the problem in case of general cooperation of a small degree.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 4

Evolution–Communication

Evolution–communication P systems (abbreviated in what follows as EC P systems) are a variant of P systems allowing both rewriting rules and symport / antiport rules, thus having separated the rewriting and the communication.

4.1 Introduction

The original variant studied was with non-cooperative rules and symport / antiport of weight one.

In [59] it has been proved by simulating programmed grammars with unconditional transfer that three membranes are sufficient to generate any Turing computable set of natural numbers, or two membranes, if one catalyst is present and catalytic rules are also allowed, while EC P systems with two membranes and non-cooperative rules generate at least Parikh sets of programmed grammars without appearance checking.

In [5], by simulating programmed grammars, these results have been improved: two membranes are sufficient to generate any Turing computable set of vectors of natural numbers. Also, EC P automata were introduced and was proved that 2-membrane EC P automata with a promoter at the level of evolution rules can accept all recursively enumerable languages. Finally, it was shown that extending the system by letting the environment contain an infinite supply of objects of one type (and ignoring this object in the result) is enough to generate all Turing computable set of vectors of natural numbers by P systems with rules of just two types: $a \to bc$ and $(b, out; c, in)$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$a, b, c \in O$.

In [134] EC P systems with symport of weight two instead of antiport of weight one are studied, obtaining similar results, i.e., generating any Turing computable set of natural numbers by EC P systems with two membranes, by simulating matrix grammars. Moreover, an extension of the model to allow the evolution rules to have the target indications was introduced.

In [11] a restriction of EC P systems called proton pumping P systems is introduced: protons do not appear in evolution rules, but participate in all communication rules. It has been shown, again by simulating programmed grammars, that any Turing computable set of vectors of natural numbers can be generated by a proton pumping P system with three membranes. Also, if only one kind of proton is used and proton pumping rules have either strong or weak priority over evolution rules, then proton pumping P systems with three membranes generate at least Parikh sets of $ET0L$ languages. It has then been suggested by Francesco Bernardini that one can show by simulating register machines that proton pumping P systems (with non-cooperative rewriting rules and symport / antiport rules of weight one) with three membranes and four kinds of protons are universal, but he never published his proof.

In [7] the deterministic EC P systems were studied, showing, by simulating register machines, that for any Turing computable set of vectors of natural numbers there exist two deterministic EC P systems with three membranes, accepting it, the first one having non-cooperative evolution rules and symport / antiport rules of weight 1, the second one having non-cooperative evolution rules and symport rules of weight at most 2.

In [61] the idea from [65] is followed, no longer assuming the rules are executed in one step, but considering a property of P systems to give the result independent on the time assigned to the rules, calling it time-freeness. It was shown by simulating programmed grammars that EC P systems with two membranes and non-cooperative evolution rules with targets and symport / antiport rules of weight one generate at least Parikh images of languages generated by programmed grammars without appearance checking, while time-free EC P systems with three membranes and non-cooperative evolution rules with targets and symport / antiport rules of weight one can generate any Turing computable set of vectors of natural numbers, and replacing antiport rules of weight one by symport rules of weight two yields the same result.

In [10] the results above were improved by simulating register machines: for any Turing computable set of vectors of natural numbers time-free EC

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

P systems with two membranes and non-cooperative evolution rules without targets and symport / antiport rules of weight one generate any Turing computable set of vectors of natural numbers. Replacing antiport rules of weight one by symport rules of weight two yields the same result. Moreover, allowing evolution rules with targets or relaxing the time-free condition yields systems able to generate any recursively enumerable language.

Finally, in [6] some results on time-free proton pumping P systems were obtained. Time-free P systems with two membranes and four protons and symport / antiport rules of weight one generate any Turing computable set of vectors of natural numbers. Time-free P systems with two membranes and four protons and symport rules of weight at most two generate any Turing-computable set of vectors of natural numbers. Moreover, allowing evolution rules with targets or relaxing the time-free condition yields systems able to generate any recursively enumerable language.

It is even more surprising that one proton is already sufficient for non-time-free P systems with two membranes and either symport / antiport of weight one, or symport of weight two to generate any recursively enumerable language.

4.2 Definitions

Let us recall the formal definition of EC P systems:

Definition 4.2.1 *An evolution–communication P system of degree $m \geq 1$ is defined as*

$$\Pi = (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1, \cdots, R'_m, i_0),$$

where O is the alphabet of objects, μ is a membrane structure with m regions, labelled with $1, \cdots, m$, and $i_0 \in \{0, \cdots, m\}$ is the output region (the environment if $i_0 = 0$). Every region $i \in \{1, \cdots, m\}$ has

- *$w_i \in O^*$ – a string representing a multiset of objects from O;*
- *R_i – a finite set of evolution rules over O of the form $u \to v$, for $u \in O^+$ and $v \in O^*$ (hence without target indications associated with the objects from v);*
- *R'_i – a finite set of symport / antiport rules over O, of the forms $(u, in), (v, out), (v, out; u, in)$, for $u, v \in O^+$.*

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The evolution rules of the form $u \to v$ mean to replace u by v, in the region this rule is associated to. These rules change the objects without moving them to a different region, as opposed to the communication rules, moving the objects without changing them. The symport / antiport rules are associated to membranes, rather than to regions. Rule (u, in) means to bring u inside the membrane, rule (v, out) means to take v out of the membrane, and rule $(v, out; u, in)$ means to bring u inside the membrane in exchange for v, which is taken out of the membrane (both u in the external region and v in the internal region must be present for the rule to be applicable, and both are used by the rule if it is applied). Both evolution and communication rules are applied in parallel.

The notation $NOP_m(\alpha, tar, sym_i, anti_j)$ ($PsOP_m(\alpha, sym_i, anti_j)$, $LOP_m(\alpha, sym_i, anti_j)$), for $\alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\}$, is used for the family of natural number sets (the family of natural number vector sets, the family of languages) generated by EC P systems with at most m membranes, using symport rules of weight at most i, antiport rules of weight at most j, and cooperative evolution rules (coo), non-cooperative rules ($ncoo$), or catalytic rules with at most k catalysts (cat_k) with targets (if evolution rules have no targets, tar is omitted).

4.3 Universality

We recall that every recursively enumerable language can be generated by a register machine with 2 registers and an output tape.

4.3.1 Symport / Antiport of Weight One

Theorem 4.3.1 $LOP_2(ncoo, sym_1, anti_1) = RE$.

Proof. We only prove the inclusion $\supseteq$. Consider an arbitrary recursively enumerable language $L \subseteq T^*$. Then there is a register machine $M = (2, T, l_0, l_h, I)$ generating L, and let $I_- = \{l \mid l : (S(i), l', l'') \in I\}$.

We will construct a P system Π simulating M in such a way that the value of register i is represented by the multiplicity of the object a_i in the skin region. The proton D_i will be used to decrement the value of register i,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

while E_i will be used to check if the register i is null.

$$\begin{aligned}
\Pi &= (O, \mu = [_1\, [_2\ ]_2\ ]_1, w_1, w_2, R_1, R_2, R'_1, R'_2), \text{ where} \\
O &= T \cup \{a_i \mid 1 \leq i \leq 2\} \cup \{l_j \mid l \in I_-, 1 \leq j \leq 4\} \\
&\cup \{\#_1, \#_2\} \cup I \cup \{D_i, E_i \mid 1 \leq i \leq 2\}, \\
w_1 &= l_0 D_1 D_2 Z_1 Z_2 \#_1,\ w_2 = \lambda,
\end{aligned}$$

and the set of the rules is the following:

For each instruction $l : (A(i), l', l'') \in I$,

- $l \to a_i l'$, $l \to a_i l'' \in R_1$.

For each instruction $l : (write(a), l', l'') \in I$ we have the rules

- $l \to a l'$, $l \to a l'' \in R_1$.
- $(a, out) \in R'_1$.

For each instruction $l : (S(i), l', l'') \in I$,

- $(l, in) \in R'_2$,

(decrement)

- $l \to l_4 \in R_2$,
- $(l_4, out; D_i, in)$, $(D_i, out; a_i, in)$, $(D_i, out; \#_1, in) \in R'_2$,
- $l_4 \to l' \in R_1$,

(zero test)

- $l \to l_1$, $l_2 \to l_3 \in R_2$,
- $(E_i, in; l_1, out)$, $(E_i, out; a_i, in)$, $(E_i, out; l_2, in)$, $(l_3, out) \in R'_2$,
- $l_1 \to l_2 \in R_1$, $l_2 \to \#_2$, $l_3 \to l'' \in R_1$.

Finally, we also have the rules

- $\#_1 \to \#_1 \in R_2$,
- $\#_2 \to \#_2 \in R_1$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

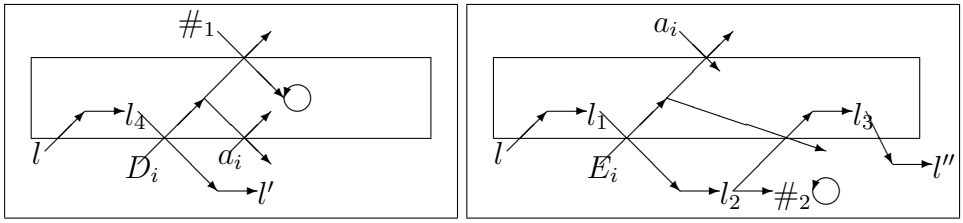


Figure 4.1: Using $(ncoo, sym_1, anti_1)$ decrement: left, zero-test: right

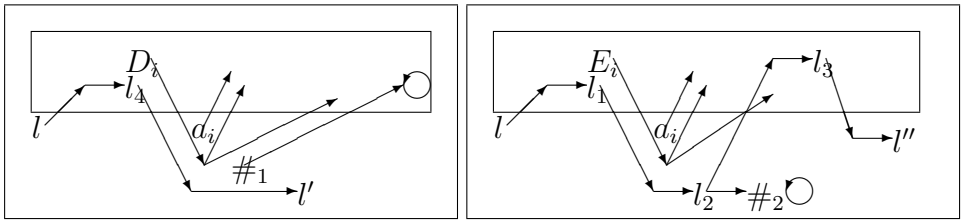


Figure 4.2: Using $(ncoo, sym_2)$ decrement: left, zero-test: right

The system constructed in that way simulates the corresponding register machine. The increment instructions are simulated in one step: the instruction symbol changes to a symbol corresponding to the next instruction and a symbol corresponding to the register being incremented.

Decrement: l comes to region 2, changes to l_4 and returns to region 1, bringing D_i to region 2, and then changes to l'. The "duty" of D_i is to decrement register i by returning to region 1 and removing one copy of a_i from region 1. If register i is null, then D_i exchanges with $\#_1$ and the computation never halts (if decrement is possible, D_i can still exchange with $\#_1$, but this case is not productive).

Zero-test: after l has come to region 2, it changes to l_1 and returns to region 1, bringing E_i to region 2, and then changes to l_2. The "duty" of E_i is to check that register i is null by waiting for l_2. If register i is not null, then E_i will immediately exchange with a_i and then l_2 will change to $\#_2$, so the computation will never halt (if E_i waits for l_2, l_2 can still change to $\#_2$, but this case is not productive).

The decrement and zero-test are illustrated in Figure 4.1. □

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

4.3.2 Symport of Weight Two

Theorem 4.3.2 $LOP_2(ncoo, sym_2) = RE$.

Proof. This is a "dual" theorem: the simulation of a register machine is done in exactly the same way, except that the protons that were in region 1 are now in region 2, and vice-versa.

$$\begin{aligned} \Pi &= (O, \mu = [_1\, [_2\ ]_2\,]_1, w_1, w_2, R_1, R_2, R'_1, R'_2), \text{ where} \\ O &= T \cup \{a_i \mid 1 \le i \le 2\} \cup \{l_j \mid l \in I_-, 1 \le j \le 4\} \\ &\cup\ \{\#_1, \#_2\} \cup I \cup \{D_i, E_i \mid 1 \le i \le 2\}, \\ w_1 &= l_0\#_1,\ w_2 = D_1D_2Z_1Z_2, \end{aligned}$$

and the set of the rules is the following:

For each instruction $l : (A(i), l', l'') \in I$,

- $l \to a_i l'$, $l \to a_i l'' \in R_1$.

For each instruction $l : (write(a), l', l'') \in I$ we have the rules

- $l \to a l'$, $l \to a l'' \in R_1$,
- $(a_i, out) \in R'_1$.

For each instruction $l : (S(i), l', l'') \in I$,

- $(l, in) \in R'_2$,

(decrement)

- $l \to l_4 \in R_2$,
- $(l_4 D_i, out)$, $(a_i D_i, in)$, $(\#_1 D_i, in) \in R'_2$,
- $l_4 \to l' \in R_1$,

(zero test)

- $l \to l_1$, $l_2 \to l_3 \in R_2$
- $(l_1 E_i, out)$, $(a_i E_i, in)$, $(l_2 E_i, in)$, $(l_3, out) \in R'_2$,
- $l_1 \to l_2 \in R_1$, $l_2 \to \#_2$, $l_3 \to l'' \in R_1$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Finally, we also have the rules

- $\#_1 \to \#_1 \in R_2$,
- $\#_2 \to \#_2 \in R_1$.

The system constructed above simulates the corresponding register machine. The increment instructions are simulated in one step: the instruction symbol changes to a symbol corresponding to the next instruction and a symbol corresponding to the register being incremented.

Decrement: l comes to region 2, changes to l_4 and returns to region 1 with D_i, and then changes to l'. The "duty" of D_i is to decrement register i by returning to region 2 and removing one copy of a_i from region 1. If register i is null, then D_i exchanges with $\#_1$ and the computation never halts (if decrement is possible, D_i can still exchange with $\#_1$, but this case is not productive).

Zero-test: after l has come to region 2, it changes to l_1 and returns to region 1 with E_i, and then changes to l_2. The "duty" of E_i is to check that register i is null by waiting for l_2. If register i is not null, then E_i will immediately exchange with a_i and then l_2 will change to $\#_2$, so the computation will never halt (if E_i waits for l_2, l_2 can still change to $\#_2$, but this case is not productive). Figure 4.2 illustrates decrementing and zero-test. □

4.4 EC P Automata: Accepting Languages

We are now going to prove that EC P automata accept all recursively enumerable languages with the help of promoters at the level of evolution rules.

Let us consider a language $L \in V^*, V = \{c_1, \cdots, c_k\}$. Every word $w = c_{m_1} \cdots c_{m_{|w|}} \in L$ corresponds to the number $val_{k+1}(w)$, which is the value of $m_1 \cdots m_{|w|}$ in base $k+1$ (a number, whose digits are the subscripts m_i of letters c_{m_i} of w). If $L \in RE$, then $val_{k+1}(L) \in NRE$, hence the following holds.

Lemma 4.4.1 *Let $L \in V^*$ and $L' = \{a^{val_{k+1}(w)} \mid w \in L\}$ for a given symbol a. If $L \in RE$, then L' is a one-letter RE language.*

Theorem 4.4.1 $A_I OP_2(p_1 ncoo, sym_1, anti_1) = RE$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Proof. Let us consider a language $L \in V^*, V = \{c_1, \cdots, c_k\}$. Let us define L' as in lemma above, and let M be a register machine which generates L'.

We construct the EC P automaton

$$\Pi_1 = (O \cup O', V, \mu, w_1 w_1', w_2 w_2', R_1 \cup Q_1, R_2 \cup Q_2, Q_1', R_2' \cup Q_2'),$$

where O, w_1, w_2, R_1, R_2, R_2' are as in the proof of Theorem 4.3.1, and

$$\begin{aligned}
O' &= V \cup \{b, b', d, e, e', g, g', I\},\\
\mu &= [_1\, [_2\ ]_2\,]_1,\\
w_1' &= I,\ w_2' = g,\\
Q_1 &= \{c_i \to b^i e|_e,\ c_i \to b^i e' g'|_e,\ c_i \to \#, b \to b^{k+1}|_e\}\\
&\cup\ \{e \to f',\ b \to b'|_{e'},\ a \to a,\ I \to e,\ I \to g'\},\\
Q_2 &= \{b' \to d,\ d \to d, g \to g\},\\
Q_1' &= \{(c_i, in) \mid c_i \in V\},\\
Q_2' &= \{(b', in),\ (d, out; a, in),\ (g, out; g', in)\}.
\end{aligned}$$

The system Π_1 has all the elements and all the rules of the system Π in the previous theorem, except the rule (a, out) is no longer assigned to the skin membrane. This part of Π_1 simulates the register machine M of L' in the way described in Theorem 4.3.1, except the terminal symbols do not leave the system. The other objects used are: the alphabet $\{c_i \mid 1 \le i \le k\}$ of the input language, I is the initializer, b is used for calculating a unary representation of the input word, b' transports this number to the inner membrane, and d stores it. Objects e and e' control the processing of the input (they are used as promoters), g and g' are used to make sure that the computation is blocked if the input word is not decoded correctly.

Suppose Π_1 is processing the input w. Rules 4 and 1 from Q_1 enforce calculation of $val_{k+1}(w)$, performing multiplication and addition, respectively. Rule 2 from Q_1 does the last addition. Rule 3 from Q_1 blocks the computation if the rule 2 from Q_1 was applied before the last input letter. Rule 5 from Q_1 erases e in the outer region, rule 6 from Q_1 stops the computation of $v = val_{k+1}(w)$, rule 7 from Q_1 keeps the terminals a busy. The initializer I makes a guess and either produces e by rule 8 from Q_1 to process the input, or g' by rule 9 from Q_1 to examine the case of the empty word. If it produces e and the word is empty, then the system never halts because of the rule 3 from Q_2 (g cannot escape by rule 3 from Q_2' because neither rule 2 nor rule

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

9 from Q_1 were applied to produce g'). If I produces g' and the word is not empty, then there is no promoter e for the rules 1, 2 from Q_1, and the input symbol will block the computation by rule 3 from Q_1.

In region 2, rule 1 of Q_2 transforms $(b')^v$ into d^v; rule 2 of Q_2 keeps d's busy, and rule 3 of Q_2 requires that g is brought into the outer region by rule 3 of Q_2' when (and only if) the value v is calculated correctly.

Rule 1 of Q_1' brings the symbols of the input string into region 1. rule 1 of Q_2' transports $(b')^v$ into the region 2; rule 2 of Q_2' is used for decrementing multiplicities of d in region 2 and of a in region 1, and the decrementation is used for comparing the numbers.

We will now proceed to a more detailed explanation of why this construction works. At first, we discuss the computation of v. In case of the empty word, $v = 0$ as described in the work of the initializer. Assume $|w| > 0$. Let $v_j = val_{k+1}(c_{m_1} \cdots c_{m_j})$. Then $v_{j+1} = v_j(k+1) + m_{j+1}$, $j < |w|$. Two "wrong" things can possibly happen in simulation: if rule 2 of Q_1 is applied not for the last symbol of w, then the computation will be blocked by rule 3 of Q_1; also if rule 2 of Q_1 is never applied, then g' will never appear, and rule 3 of Q_2 will block the computation. These cases produce no results.

After all the input is successfully converted to v objects b, the symbols e' and g' appear. The first one promotes b^v to $(b')^v$ by rule 6 of Q_1, the latter removes g from the inner region by rule 3 of Q_2'. Then, $(b')^v$ come to the inner region by rule 1 of Q_2' and transform into d^v by rule 1 of Q_2.

Also, the derivation of M is simulated, producing some number x of objects a in the outer region. The terminal configuration can be achieved for any (and only for such) v, that $x = v$ can be produced (without blocking). The comparison is done by rule 2 of Q_2'. If the numbers are not equal, then either rule 7 of Q_1 or rule 2 of Q_2 produces an endless computation.

This way, Π_1 accepts the language $L(G)$ and the theorem statement follows immediately by Lemma 2.2.1. □

The following "dual" result, not published anywhere, is not so surprising: antiport rules can be replaced by cooperative symport rules.

Theorem 4.4.2 $A_I OP_2(p_1 ncoo, sym_2) = RE$.

Proof. Let us again consider a language $L \in V^*, V = \{c_1, \cdots, c_k\}$, define L' as in lemma above, and let M be a register machine which generates L'.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

We construct the EC P automaton

$$\Pi_2 = (O \cup O', V, \mu, w_1 w_1', w_2, R_1 \cup Q_1, R_2, Q_1', R_2' \cup Q_2'),$$

where O, w_1, w_2, R_1, R_2, R_2' are as in the proof of Theorem 4.3.2, and

$$\begin{aligned}
O' &= V \cup \{b, b', e, e', g, g', I\},\\
\mu &= [_1\, [_2\ ]_2\,]_1,\\
w_1' &= Ig,\\
Q_1 &= \{c_i \to b^i e|_e,\ c_i \to b^i e' g'|_e,\ c_i \to \#, b \to b^{k+1}|_e\}\\
&\cup\ \{e \to f',\ b \to b'|_{e'},\ a \to a,\ I \to e,\ I \to g'\}\\
&\cup\ \{b' \to b', g \to g\},\\
Q_1' &= \{(c_i, in) \mid c_i \in V\},\\
Q_2' &= \{(b'a, in),\ (gg', in)\}.
\end{aligned}$$

The system Π_2 has all the elements and all the rules of the system Π in Theorem 4.3.2, except the rule (a, out) is no longer assigned to the skin membrane. The computation of M is done like in Theorem 4.3.2, except a stays in the skin. The encoding of the input in unary is done exactly like in the previous theorem, but objects b' do not automatically move to the inner region and do not change into objects d. Instead of comparing the multiplicities of objects a and d, we directly compare the multiplicities of objects a and b', that otherwise "wait to be compared" (see rules $a \to a$ and $b' \to b'$), by moving them to region 2. The system can halt if and only if the multiplicities coincide, and all objects a and b' are removed from the skin region.

This way, Π_2 accepts the language $L(G)$ and the theorem statement follows immediately by Lemma 2.2.1. □

4.5 Infinite Environment

Let us now extend the alphabet of an EC P system by a distinguished object which we call "water"; water is present in the environment in an unbounded quantity, and can be brought inside the skin membrane in exchange for the output (in the non-extended model, all communications with the environment were limited to outputting the result). This simple change allows us

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

to obtain universality without using symport rules of weight one and with binary evolution rules, that is, rules of the form $a \to bc$. We denote the corresponding families of languages by $E_1OP_m(ncoo_2, sym_i, anti_j)$, with the obvious meaning (E_1 stands for extending the alphabet by one object).

Theorem 4.5.1 $LE_1OP_2(ncoo_2, anti_1) = RE$,
$LE_1OP_2(ncoo_2, sym_{=2}) = RE$.

Proof. (sketch) Let us denote water symbol by \$. We will use the construction from Theorem 4.3.1 to prove the first statement, and the one from Theorem 4.3.2 to prove the latter statement. Notice that both constructions all evolution rules have either 1 or 2 symbols in the right-hand side. Replace all rules $a \to b$ by $a \to b\$$, all rules (a, in) by $(\$, out; a, in)$ or by $(a\$, in)$ and all rules (a, out) by $(a, out; \$, in)$ or by $(a\$, out)$ (take the first case of each or clause for the first claim, the second case of each or clause of the second claim). Finally, make sure there is enough water for the computation (namely, for simulating symport rules of weight 1) in all regions.

Notice that the only such rules that act across membrane 2 in theorems 4.3.1 and 4.3.2 are (l, in), (l_3, out) that happen at most once per direction in simulation of a SUB instruction, however, renaming rules take place in both regions every time a SUB instruction is simulated, so it is already sufficient to add one object \$ to every region.

The only symport rules of weight 1 acting across membrane 1 in theorems 4.3.1 and 4.3.2 are (a_i, out). This is fine for the antiport model (since we assume that the environment has an unbounded supply of water), while for the pure symport model we could add to the alphabet O symbols l_1 associated to ADD instructions l, and replace rules $l \to a_i l'$, $l \to a_i l''$ from R_1 by rules $l \to l_1\$$, $l_1 \to a_i l'$, $l_1 \to a_i l''$, which proves the theorem. □

4.6 Time-Freeness

Given a time-mapping

$$e : R_1 \cup R_2 \cup \cdots \cup R_m \cup R'_1 \cup R'_2 \cup \cdots \cup R'_m \longrightarrow \mathbb{N}$$

and an EC P system Π as defined above, it is possible to construct a *timed EC P system* $\Pi(e)$ as $(O, \mu, w_1, w_2, \ldots, w_m, R_1, \ldots, R_m, R'_1, \ldots, R'_m, i_0, e)$ working in the following way.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

We suppose the existence of an external and global clock that ticks at uniform intervals of time. At each time in the regions of the system we have together rules (both evolution and transport) in execution and rules not in execution. At each time all the evolution and transport rules that can be applied (started) in each region, have to be applied. If a rule $r \in R_i, R'_i, 1 \leq i \leq m$, is applied, then all objects that can be processed by the rule have to evolve by this rule (a rule is applied in a maximally parallel manner as standard in P system area).

As usual, the rules from R_i are applied to objects in region i and the rules from R'_i govern the communication of objects through membrane i. There is no difference between evolution rules and communication rules: they are chosen and applied in the non-deterministic maximally parallel manner. When an evolution rule or a transport rule r is started at time j, its execution terminates at time $j + e(r)$. If two rules are started in the same time unit, then possible conflicts for using the copies of objects are solved assigning the objects in a non-deterministic way (again, in the way usually defined in P system area). Notice that when the execution of a rule r is started, the occurrences of objects used by this rule are not anymore available for other rules during the entire execution of r.

The computation stops when no rule can be applied in any region and there are no rules in execution: in this case the system has reached a *halting configuration*. The output of a halting computation is the vector of numbers representing the multiplicities of object presents in the output region in the halting configuration. (If $i_0 = 0$, then also the sequence of objects sent outside can be considered as the result; in this case, if some objects arrive into the environment simultaneously, then every permutation is considered.) Collecting all the vectors obtained, for any possible halting computation, we get the set of vectors of natural numbers generated by the system. (If we collect the sequences of objects, then we obtain a language.)

An EC P system $\Pi = (O, \mu, w_1, w_2, \ldots, w_m, R_1, \ldots, R_m, R'_1, \ldots, R'_m, i_0)$ *is* time-free *if and only if every system in the set*

$$\{\Pi(e) \mid e : R \longrightarrow \mathbb{N}\}$$

(where $R = R_1 \cup R_2 \cup \cdots \cup R_m \cup R'_1 \cup R'_2 \cup \cdots \cup R'_m$*) produces the same set of vectors of natural numbers (or the same language).*

Because there is no ambiguity, in this case the set of vectors of natural number generated by a time-free *EC* P system Π is indicated by $Ps(\Pi)$ (the

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

corresponding language generated is denoted by $L(\Pi)$). We use the notation $fPsOP_m(ncoo, sym_i, anti_j)$ to denote the family of sets of vectors of natural numbers generated by *time-free EC* P systems with at most m membranes (as usually, $m = *$ if such a number is unbounded), non-cooperative evolution rules, symport rules of weight at most i, and antiport rules of weight at most j. If languages are generated, then we replace Ps by L in the notation.

Notice that P systems constructed in theorems 4.3.1 and 4.3.2 are actually time-free (in the sense of generating $PsRE$).

Corollary 4.6.1 $fPsOP_2(ncoo, sym_1, anti_1) = fPsOP_2(ncoo, sym_2) = PsRE$.

In the time-free systems, we have no way of controlling the order in which objects $a \in T$ exit the system, and the order cannot be enforced because nothing else should be sent into the environment except the result. However, sending objects a directly to the environment using targets by the rules producing them will yield a similar result for the time-free EC P systems with targets.

Corollary 4.6.2 $fLOP_2(ncoo, tar, sym_1, anti_1) = fLOP_2(ncoo, tar, sym_2) = RE$.

4.7 Determinism

4.7.1 Symport / Antiport of Weight One

Theorem 4.7.1 $DN_aOP_3(ncoo, sym_1, anti_1) = NRE$.

Proof. Given a set $M \in NRE$, consider a deterministic register machine $G = (m, e_{init}, e_{halt}, P)$ with m registers, initial label e_{init}, halting label e_{halt}, instruction set P, and set $Lab(P)$ of labels, accepting M. We construct the following P system (object a_i represents the the value of the i-th register of G).

$$\begin{aligned}
\Pi &= (O, \mu = [_1\, [_2\, [_3\ ]_3\ ]_2\ ]_1, w_1 = \lambda, w_2 = e_{init}, w_3 = \lambda, \\
&\quad R_1, R_2, R_3, R_1' = \emptyset, R_2', R_3', 2), \\
O &= \{e, e_0, e_1, e_2, e_3, e_4, e_5, e_6 \mid e \in Lab(P)\} \\
&\cup \{a_r \mid 1 \leq r \leq m\} \cup \{s_1, s_2, s_3, q\},
\end{aligned}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$
\begin{aligned}
R_1 &= \{s_1 \to s_2\} \cup \{a_r \to \lambda \mid 1 \le r \le m\} \\
&\cup \{e_3 \to e_4 \mid e : (S(r), f, g) \in P\}, \\
R_2 &= \{s_3 \to \lambda\} \cup \{e \to a_r f \mid e : (A(r), f) \in P\} \\
&\cup \{e \to s_1 e_0,\ e_0 \to e_1,\ e_1 \to e_2,\ e_2 \to e_3,\ e_4 \to e_5 q,\ e_5 \to e_6,\ e_6 \to g \\
&\quad \mid e : (S(r), f, g) \in P\}, \\
R_3 &= \{s_2 \to s_3, q \to \lambda\} \cup \{e_3 \to f \mid e : (S(r), f, g) \in P\}, \\
R'_2 &= \{(s_1, out),\ (a_r, out; s_2, in)\} \\
&\cup \{(e_3, out; s_2, in),\ (e_4, in) \mid e : (S(r), f, g) \in P\}, \\
R'_3 &= \{(s_2, in),\ (s_3, out; q, in)\} \cup \{(f, out) \mid e : (S(r), f, g) \in P\} \\
&\cup \{(s_3, out; e_3, in) \mid e : (S(r), f, g) \in P\}.
\end{aligned}
$$

The P system above recognizes a number N if and only if the computation, starting with a_1^N (the input register of G is the first one) placed in region 2, halts. Below are the simulations of individual instructions.

Instruction $e : (A(r), f)$ is simulated in the following way:

$$[_1\ [_2\ ew[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ a_r f w[_3\ ]_3\ ]_2\ ]_1.$$

The object e (corresponding to the instruction label) simply evolves into $a_r f$, thus changing instruction label from e to f and adding one to the counter r.

Simulation of instruction $e : (S(r), f, g)$ (in case register r is non-zero):

$$
\begin{aligned}
& [_1\ [_2\ ea_r w[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_1 e_0 a_r w[_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ s_1[_2\ e_1 a_r w[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ s_2[_2\ e_2 a_r w[_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ a_r[_2\ s_2 e_3 w[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ e_3 w[_3\ s_2\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ [_2\ e_3 w[_3\ s_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_3 w[_3\ e_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ [_2\ w[_3\ f\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ fw[_3\ ]_3\ ]_2\ ]_1.
\end{aligned}
$$

The object e (corresponding to the instruction label) evolves into e_0 (changing in 3 steps into e_3) and s_1, which goes in region 1, then changes into s_2, and then returns in region 2 in exchange for a_r (which is then erased). Then, s_2 travels into region 3, changes to s_3 and returns to region 2 (where it is then erased) in exchange for e_3. Finally, e_3, being in region 3, changes into f and return in region 2, finishing the simulation of the instruction.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Instruction $e : (S(r), f, g)$ (in case register r is zero) is simulated as follows:

$$\begin{aligned}
& [_1\ [_2\ ew[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_1e_0rw[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ s_1[_2\ e_1w[_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ s_2[_2\ e_2w[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ s_2[_2\ e_3w[_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ e_3[_2\ s_2w[_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ e_4[_2\ w[_3\ s_2\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ e_4w[_3\ s_3\ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ [_2\ e_5qw[_3\ s_3\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ e_6ws_3[_3\ q\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ gw[_3\ ]_3\ ]_2\ ]_1.
\end{aligned}$$

(Note that $|w|_{a_r} = 0$.) Like in the previous case, the object e evolves into e_0 (changing in 3 steps into e_3) and s_1, which goes in region 1, and then changes into s_2. Now there is no object a_r in region 2 to bring s_2 to region 2, so s_2 remains in region 3 until the next step, when it is exchanged with e_3. Then s_2 travels to region 3 and changes into s_3. Now, e_3, being in region 1, changes into e_4, returns to region 2, where it evolves into e_5 (changing it two steps into g) and q, which exchanges with s_3 and then both q and s_3 are erased.

These two cases are graphically represented in Figure 4.3. □

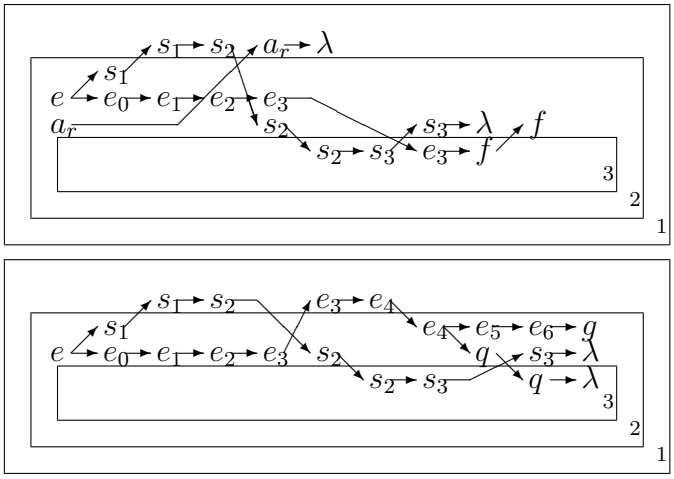


Figure 4.3: Deterministic P systems with $(ncoo, sym_1, anti_1)$. Decrement (top) and zero-test (bottom).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

4.7.2 Symport of Weight One

In the next theorem, symport of weight two is used instead of antiport of weight one, leading to one more universality result.

Theorem 4.7.2 $DN_aOP_3(ncoo, sym_2) = NRE$.

Proof. Given a set $M \in NRE$, consider a deterministic register machine $G = (m, e_{init}, e_{halt}, P)$ as above, accepting M. We construct the following P system:

$$
\begin{aligned}
\Pi &= (O, \mu = [_1\ [_2\ [_3\ ]_3\ ]_2\ ]_1, w_1 = \lambda, w_2 = e_{init}, w_3 = \lambda,\\
&\quad R_1, R_2, R_3, R_1' = \emptyset, R_2', R_3', 2),\\
O &= \{e, e_0, e_1, e_2, e_3, e_4, e_5, e_6 \mid e \in Lab(P)\}\\
&\cup \{a_r \mid 1 \le r \le m\} \cup \{s_1, s_2, s_3, q\},\\
R_1 &= \{q \to \lambda, s_2 \to \lambda\} \cup \{e_0 \to e_1 \mid e : (S(r), f, g) \in P\}\\
&\cup \{a_r \to \lambda \mid 1 \le r \le m\},\\
R_2 &= \{s_1 \to s_2\} \cup \{e \to s_1 e_0,\ e_1 \to e_2 q,\ e_2 \to e_3,\ e_3 \to f\\
&\quad \mid e : (S(r), f, g) \in P\}\\
&\cup \{e \to a_r f \mid e : (A(r), f) \in P\},\\
R_3 &= \{s_2 \to \lambda\} \cup \{e_0 \to g \mid e : (S(r), f, g) \in P\},\\
R_2' &= \{(q s_2, out)\} \cup \{(e_0 a_r, out),\ (e_1, in) \mid e : (A(r), f) \in P\},\\
R_3' &= \{(s_2 e_0, in),\ (g, out) \mid e : (S(r), f, g) \in P)\}.
\end{aligned}
$$

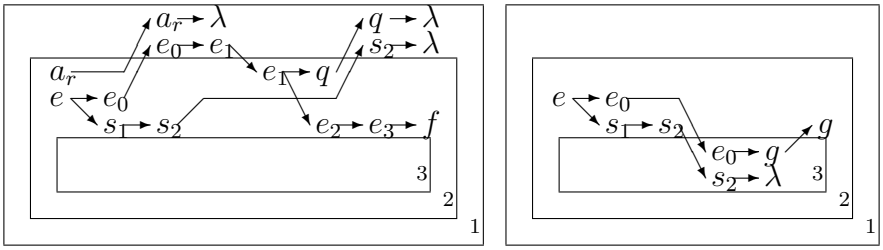


Figure 4.4: Deterministic P systems with $(ncoo, sym_2)$. Decrement (left) and zero-test (right).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The P system above recognizes a number N if and only if the computation, starting with a_1^N (the first register is the input one of G) placed in region 2, halts. Below is the simulation of the instructions of G.

Instruction $e : (A(r), f)$ is simulated like in the previous theorem:

$$[_1\ [_2\ ew[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ Rfw[_3\ \ ]_3\ ]_2\ ]_1.$$

Instruction $e : (S(r), f, g)$ is simulated in the following way: The object e evolves in e_0 (used to subtract) and s_1 (which changes into s_2, the helper).

$$\begin{aligned}
& [_1\ [_2\ ea_rw[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_1e_0a_rw[_3\ \ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ e_0a_r[_2\ s_2w[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ e_1[_2\ s_2w[_3\ \ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ [_2\ e_1s_2w[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ e_2qs_2w[_3\ \ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ qs_2[_2\ e_3w[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ fw[_3\ \ ]_3\ ]_2\ ]_1.
\end{aligned}$$

If a_r is present in region 2, then (one copy of) a_r goes to region 1 (where it is erased) together with e_0, which changes into e_1, returns to region 2, and then evolves into e_2 (which changes into f in two steps) and q, which exists to region 1 together with s_2, where both q and s_2 are erased.

$$\begin{aligned}
& [_1\ [_2\ ew[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_1e_0w[_3\ \ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ s_2e_0w[_3\ \ ]_3\ ]_2\ ]_1 \\
\Rightarrow\ & [_1\ [_2\ w[_3\ s_2e_0\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ w[_3\ g\ ]_3\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ gw[_3\ \ ]_3\ ]_2\ ]_1.
\end{aligned}$$

(Note that $|w|_{a_r} = 0$.) If a_r is not present in region 2, then e_0 waits for s_2, they both come to region 3, where s_2 is erased, while e_0 changes to g and returns to region 2, finishing the simulation of the instruction.

These two cases are graphically represented in Figure 4.4. □

4.8 Proton Pumping

Definition 4.8.1 *A proton pumping P system of degree $m \geq 1$ is defined as*

$$\Pi = (O, P, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1, \cdots, R'_m, i_0), \tag{4.1}$$

where

$$(O, \mu, w_0, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1 \cdots, R'_m, i_0)$$

is an evolution–communication P system, $P \subseteq O$ is the set of protons, and all rules are of the following forms:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- $u \to v$, *where* $u \in (O-P)^+, v \in (O-P)^*$ *(evolution rule, does not involve protons).*
- (a, out) *or* (a, in), *where* $a \in O$ *(uniport rule),*
- (x, out) *or* (x, in), *where* $x \in O^+$, $|x|_P = 1$ *(proton pumping symport rule),*
- $(x, out; y, in)$, *where* $x, y \in O^+$ *and* $|xy|_P = 1$ *(proton pumping antiport rule).*

Thus, proton pumping P systems are a restricted variant of EC P systems.

We use the following notations

$$\mathbf{X}ProP^{\mathbf{k}}_{\mathbf{m}}(ncoo, tar, sym_{\mathbf{i}}, anti_{\mathbf{j}})$$

to denote the family of languages ($\mathbf{X} = L$), vector sets ($\mathbf{X} = Ps$) or number sets ($\mathbf{X} = N$) generated by proton pumping P systems with at most $\mathbf{m}$ membranes, $\mathbf{k}$ different types of protons (i.e., $\mathbf{k}$ is the cardinality of the set P), using symport rules of weight at most $\mathbf{i}$, antiport rules of weight at most $\mathbf{j}$, and non-cooperative evolution rules with targets. If targets are not allowed, then *tar* is removed from the notation (like any other unused feature). If one of the numbers $\mathbf{m}, \mathbf{k}, \mathbf{i}, \mathbf{j}$ is unbounded, we write $*$ instead).

Once again we recall theorems 4.3.1 and 4.3.2 proving the computational completeness of EC P systems. Looking into the proofs, one can easily see that the constructed P systems are actually proton pumping P systems with four protons, namely $\{D_i, E_i \mid 1 \leq i \leq 2\}$.

Corollary 4.8.1 $LProP^4_2(ncoo, sym_2, anti_1) = LProP^4_2(noo, sym_2) = RE.$

In the proofs of the preceding theorems the output symbols are generated in the right order; however, generating languages by these constructions is not time-free because the different execution times of the rules sending output symbols to the environment might lead to changing the order of symbols in the output word.

Corollary 4.8.2 $fPsProP^4_2(ncoo, sym_2, anti_1) = fPsProP^4_2(noo, sym_2) = PsRE.$

Nevertheless, if target indications are allowed, then, replacing rules $l \to al' \in R_1$, $l \to al'' \in R_1$, $(a, out) \in R'_1$ for $a \in T$ by $l \to a_{out}l' \in R_1$, $l \to a_{out}l'' \in R_1$, one obtains time-free P systems generating RE.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Corollary 4.8.3 $fLProP_2^4(ncoo, tar, sym_1, anti_1) = RE$, $fLProP_2^4(ncoo, tar, sym_2) = RE$.

4.9 One Proton

We will now show that if time-freeness is not required, then even one proton is enough for computational completeness, again with only two membranes.

4.9.1 Symport / Antiport of Weight One

Theorem 4.9.1 $LProP_2^1(ncoo, sym_1, anti_1) = RE$.

Proof. We only prove the inclusion $\subseteq$. Consider an arbitrary recursively enumerable language $L \subseteq T^*$. Then there is a register machine $M = (2, T, l_0, l_h, I)$ generating L.

We will construct a P system Π simulating M in such a way that the value of register i is represented by the multiplicity of the object a_i in region i. The proton p will be used to decrement/zero test the value of the working registers.

$$\begin{aligned}
\Pi &= (O, P, \mu = [_1\ [_2\ ]_2\ ]_1, w_1, w_2, R_1, R_2, R_1', R_2'), \text{ where} \\
O &= T \cup \{a_i, a_i' \mid 1 \leq i \leq 2\} \cup \{l_j \mid l \in I, 1 \leq j \leq 9\} \cup I \cup P \\
&\cup \{\#, I_{2,4}, I_{1,3}, I_{0,2}\} \cup \{I_j \mid 0 \leq j \leq 2\} \cup \{O_j \mid 0 \leq j \leq 5\}, \\
P &= \{p\},\ w_1 = pI_1,\ w_2 = o_0 l_0,
\end{aligned}$$

and the set of the rules is the following:

Rules related to special objects which "wait" for a certain time and then must exchange with the proton (or else the trap symbol will be introduced):

- $I_{2,4} \to I_{1,3},\ I_{1,3} \to I_{0,2} \in R_2$,
- $(I_{0,2}, out) \in R_2'$,
- $I_{0,2} \to I_0 I_2 \in R_1$,
- $I_{j+1} \to I_j \in R_1,\ 0 \leq j \leq 1$,
- $(p, out; I_0, in) \in R_2'$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Instruction	Decrement a_1		Zero-test a_1	
Step	Region 1	Region 2	Region 1	Region 2
1	$a_1 I_1 p$	$l O_0$	$I_1 p$	$l O_0$
2	$a_1 I_0 O_0$	$l_1 I_{2,4} O_5 O_3 O_1 p$	$I_0 O_0$	$l_6 I_{1,3} O_4 O_1 p$
3	$a_1 p$	$l_2 I_{1,3} O_4 O_2 O_0 I_0$	p	$l_7 I_{0,2} O_3 O_0 I_0$
4	$a_1 O_0$	$l_3 I_{0,2} O_3 O_1 p$	$I_{0,2} O_0$	$l_8 O_2 p$
5	$I_{0,2} p$	$a_1 l_4 O_2 O_0$	$I_2 I_0$	$l_9 O_1 p$
6	$I_2 I_0 O_0$	$a_1 l_5 O_1 p$	$I_1 p$	$l'' O_0 I_0$
7	$I_1 p$	$a_1 l' O_0 I_0$	Next instr.	Next instr.

Instruction	Decrement a_2		Zero-test a_2	
Step	Region 1	Region 2	Region 1	Region 2
1	$I_1 p$	$a_2 l O_0$	$I_1 p$	$l O_0$
2	$I_0 O_0$	$a_2 l_1 I_{0,2} O_3 p$	$I_0 O_0$	$l_6 I_{1,3} O_4 O_2 p$
3	$I_{0,2} p$	$a_2 l_2 O_2 I_0$	p	$l_7 I_{0,2} O_3 O_1 I_0$
4	$a_2 I_2 I_0$	$l_3 O_1 p$	$I_{0,2} p$	$l_8 O_2 O_0$
5	$a_2 I_1 p$	$l' O_0 I_0$	$I_2 I_0 O_0$	$l_9 O_1 p$
6	Next instr.	Next instr.	$I_1 p$	$l'' O_0 I_0$

Table 4.1: Proton pumping by antiport. Register operations.

- $O_0 \to \lambda$, $I_0 \to \#$, $\# \to \# \in R_1$,
- $O_{j+1} \to O_j \in R_2$, $0 \leq j \leq 4$,
- $(O_0, out; p, in) \in R'_2$,
- $I_0 \to \lambda$, $O_0 \to \#$, $\# \to \# \in R_2$.

Rules of interaction of the proton and register symbols:

- $(p, out; a_1, in)$, $(a_2, out; p, in) \in R'_2$,

For each instruction $l : (A(i), l', l'') \in I$,

- $l \to a'_i l_1 O_3 O_1 I_{0,2} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 2$,
- $l_3 \to l'$, $l_3 \to l'' \in R_2$.

The output instructions $l : (write(a), l', l'') \in I$ are simulated exactly as the addition instructions above, replacing a'_i by a.

The register symbols a_i in region i and the output symbols a in the environment are produced by the rules

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Instruction	Increment a_i/write a	
Region	1	2
1	$I_1 p$	lO_0
2	$I_0 O_0$	$l_1(a_i'$ or $a) I_{0,2} O_3 O_1 p$
3	$I_{0,2} p$	$l_2 O_2 O_0 I_0$
4	$I_2 I_0 O_0$	$l_3 O_1 p$
5	$I_1 p$	$(l'$ or $l'') O_0 I_0$s

Instruction	a		i=1	i=2	Terminate	
Region	0	1	1	2	1	2
1					pI_1	$l_h O_0$
2					$O_0 I_0$	p
3		a	a'_{m+1}	a_{m+2}	p	I_0
4	a		a_{m+1}	a_{m+2}	p	
5			a_{m+1}	a_{m+2}	Halt	Halt

Table 4.2: Proton pumping by antiport. Miscellaneous

- $a'_2 \to a_2 \in R_2$,
- (a'_1, out), $(a, out) \in R'_2$, $a \in T$,
- $a'_1 \to a_1 \in R_1$,
- $(a, out) \in R'_1$, $a \in T$.

For each instruction $l : (S(1), l', l'') \in I$,
(decrement)

- $l \to l_1 O_1 O_3 O_5 I_{2,4} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 4$,
- $l_5 \to l' \in R_2$.

(zero test)

- $l \to l_6 O_1 O_4 I_{1,3} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $6 \leq j \leq 8$,
- $l_9 \to l'' \in R_2$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

For each instruction $l : (S(2), l', l'') \in I$,
(decrement)

- $l \to l_1 O_3 I_{0,2} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 2$,
- $l_3 \to l' \in R_2$.

(zero test)

- $l \to l_6 O_2 O_4 I_{1,3} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $6 \leq j \leq 8$,
- $l_9 \to l'' \in R_2$.

For terminating the computation we have

- $l_h \to \lambda \in R_2$.

The simulation is illustrated by the tables below. Notice that every time an antiport rule is possible it must be executed, otherwise one of the objects O_0, I_0 will change to #, leading to an infinite computation.

The intuitive idea behind this construction is to create a "predefined scenario" for the proton; if the system tries to decrement a null register or the system zero-tests a non-null register, then the proton ends up in a "wrong" region and cannot follow the "scenario" anymore. We now list the scenarios for the proton, for different instructions:

- Decrement register 1: p exchanges with O_0, then with I_0, then with O_0, then with a_1, then with O_0, and finally with I_0.
- Zero-test register 1: p exchanges with O_0, then with I_0, then with O_0, then waits one step because there is no a_1, and finally with I_0.
- Decrement register 2: p exchanges with O_0, then with I_0, then with a_2, and finally I_0.
- Zero-test register 2: p exchanges with O_0, then with I_0, then waits one step because there is no a_2, then with O_0, and finally I_0.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- Increment any register or output a symbol: p exchanges with O_0, then with I_0, then with O_0, and finally I_0.

Notice that the first two steps of the simulation are always the same. This is needed to "keep the proton busy" while the object associated to the instruction creates the rest of the scenario. The scenario is created by producing objects O_0 in region 2 and objects I_0 in region 1, with corresponding delays.

When the output register is incremented, the corresponding symbol is sent to the environment, contributing to the result. At the end of the correct simulation, object l_h is erased, registers 1 and 2 are null, so no objects are present in region 2, while region 1 only contains p. □

4.9.2 Symport of Weight Two

Theorem 4.9.2 $fPsProP_2^4(ncoo, sym_2) = PsRE.$

Proof. This is a "dual" theorem: the simulation of a register machine is done in exactly the same way, except that the proton that was in region 1 is now in region 2, and vice-versa, and except that the halting is slightly modified such that the proton stays in region 1.

$$\begin{aligned}
\Pi &= (O, P, \mu = [_1\ [_2\ \ ]_2\ ]_1, w_1, w_2, R_1, R_2, R_1', R_2'), \text{ where} \\
O &= T \cup \{a_i, a_i' \mid 1 \leq i \leq 2\} \cup \{l_j \mid l \in I, 1 \leq j \leq 9\} \cup I \cup P \\
&\cup \{\#, I_{2,4}, I_{1,3}, I_{0,2}\} \cup \{I_j \mid 0 \leq j \leq 2\} \cup \{O_j \mid 0 \leq j \leq 5\}, \\
P &= \{p\},\ w_1 = I_1,\ w_2 = po_0l_0,
\end{aligned}$$

and the set of the rules is the following:

Rules related to special objects which "wait" for a certain time and then must exchange with the proton (or else the trap symbol will be introduced):

- $I_{2,4} \to I_{1,3}$, $I_{1,3} \to I_{0,2} \in R_2$,
- $(I_{0,2}, out) \in R_2'$,
- $I_{0,2} \to I_0 I_2 \in R_1$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Instruction	Decrement a_1		Zero-test a_1	
Step	Region 1	Region 2	Region 1	Region 2
1	$a_1 I_1$	lO_0p	I_1	lO_0p
2	$a_1 I_0 O_0 p$	$l_1 I_{2,4} O_5 O_3 O_1$	$I_0 O_0 p$	$l_6 I_{1,3} O_4 O_1$
3	a_1	$l_2 I_{1,3} O_4 O_2 O_0 I_0 p$		$l_7 I_{0,2} O_3 O_0 I_0 p$
4	$a_1 O_0 p$	$l_3 I_{0,2} O_3 O_1$	$I_{0,2} O_0 p$	$l_8 O_2$
5	$I_{0,2}$	$a_1 l_4 O_2 O_0 p$	$I_2 I_0 p$	$l_9 O_1$
6	$I_2 I_0 O_0 p$	$a_1 l_5 O_1$	I_1	$l'' O_0 I_0 p$
7	I_1	$a_1 l' O_0 I_0 p$	Next instr.	Next instr.

Table 4.3: Proton pumping by symport. Register 1.

Instruction	Decrement a_2		Zero-test a_2	
Step	Region 1	Region 2	Region 1	Region 2
1	I_1	$a_2 lO_0p$	I_1	lO_0p
2	$I_0 O_0 p$	$a_2 l_1 I_{0,2} O_3$	$I_0 O_0 p$	$l_6 I_{1,3} O_4 O_2$
3	$I_{0,2}$	$a_2 l_2 O_2 I_0 p$		$l_7 I_{0,2} O_3 O_1 I_0 p$
4	$a_2 I_2 I_0 p$	$l_3 O_1$	$I_{0,2}$	$l_8 O_2 O_0 p$
5	$a_2 I_1$	$l' O_0 I_0 p$	$I_2 I_0 O_0 p$	$l_9 O_1$
6	Next instr.	Next instr.	I_1	$l'' O_0 I_0 p$

Table 4.4: Proton pumping by symport. Register 2.

- $I_{j+1} \to I_j \in R_1$, $0 \leq j \leq 1$,
- $(pI_0, in) \in R'_2$,
- $O_0 \to \lambda$, $I_0 \to \#$, $\# \to \# \in R_1$,
- $O_{j+1} \to O_j \in R_2$, $0 \leq j \leq 4$,
- $(pO_0, out) \in R'_2$,
- $I_0 \to \lambda$, $O_0 \to \#$, $\# \to \# \in R_2$.

Rules of interaction of the proton and register symbols:

- (pa_1, in), $(pa_2, out) \in R'_2$,

For each instruction $l : (A(i), l', l'') \in I$,

- $l \to a'_i l_1 O_3 O_1 I_{0,2} \in R_2$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Instruction	Increment a_i/write a	
Region	1	2
1	I_1	lO_0p
2	I_0O_0p	$l_1(a_i'$ or $a)I_{0,2}O_3O_1$
3	$I_{0,2}$	$l_2O_2O_0I_0p$
4	$I_2I_0O_0p$	l_3O_1
5	I_1	l'/l'' O_0I_0p

Table 4.5: Proton pumping by symport. Miscellaneous 1.

Instruction	a		i=1	i=2	Terminate	
Region	0	1	1	2	1	2
1					I_1	l_hO_0p
2					O_0I_0p	O_1
3				a_2		I_0O_0p
4	a		a_1	a_2	O_0p	
5			a_1	a_2	p	

Table 4.6: Proton pumping by symport. Miscellaneous 2.

- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 2$,
- $l_3 \to l'$, $l_3 \to l'' \in R_2$.

The output instructions $l : (write(a), l', l'') \in I$ are simulated exactly as the addition instructions above, replacing a_i' by a.

The register symbols a_i in region i and the output symbols a in the environment are produced by the rules

- $a_2' \to a_2 \in R_2$,
- (a_1', out), $(a, out) \in R_2'$, $a \in T$,
- $a_1' \to a_1 \in R_1$,
- $(a, out) \in R_1'$, $a \in T$.

For each instruction $l : (S(1), l', l'') \in I$,
(decrement)

- $l \to l_1O_1O_3O_5I_{2,4} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 4$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- $l_5 \to l' \in R_2$.

(zero test)

- $l \to l_6 O_1 O_4 I_{1,3} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $6 \leq j \leq 8$,
- $l_9 \to l'' \in R_2$.

For each instruction $l : (S(2), l', l'') \in I$,
(decrement)

- $l \to l_1 O_3 I_{0,2} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $1 \leq j \leq 2$,
- $l_3 \to l' \in R_2$.

(zero test)

- $l \to l_6 O_2 O_4 I_{1,3} \in R_2$,
- $l_j \to l_{j+1} \in R_2$, $6 \leq j \leq 8$,
- $l_9 \to l'' \in R_2$.

For terminating the computation we have

- $l_h \to O_1 \in R_2$.

The simulation is illustrated by the tables below. Notice that every time an antiport rule is possible it must be executed, otherwise one of the objects O_0, I_0 will change to #, leading to an infinite computation.

Like in the previous proof, to arrive at a halting configuration, the proton must follow the "predefined scenario" created by instruction objects. If the system tries to decrement a null register or the system zero-tests a non-null register, then the proton ends up in a "wrong" region and cannot follow the "scenario". We now list the scenarios for the proton, for different instructions.

- Decrement a_1: p accompanies O_0, then I_0, then O_0, then a_1, then O_0, and finally I_0.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- Zero-test a_1: p accompanies O_0, then I_0, then O_0, then waits one step because there is no a_1, and finally goes with I_0.
- Decrement a_2: p moves O_0, then I_0, then a_2, and finally I_0.
- Zero-test a_2: p accompanies O_0, then I_0, then waits one step because there is no a_2, then goes with O_0, and finally with I_0.
- Increment any register or output a symbol: p accompanies O_0, then I_0, then O_0, and finally I_0.
- Halt: p accompanies O_0, then I_0, and finally O_0.

Again, the first two steps are the same, to "keep the proton busy" while the object associated to the instruction creates the rest of the scenario. The scenario is created by producing objects O_0 in region 2 and objects I_0 in region 1, with corresponding delays.

When the output register is incremented, the corresponding symbol is sent to the environment, contributing to the result. At the end of the correct simulation, object l_h changes to O_0 in 3 steps, moving p to region 1. Since registers 1 and 2 are null, no objects are present in region 2, while region 1 only contains p. □

Consider either of the theorems above. Remove the rules (a, out), $a \in T$ from R'_1 and R'_2. The output of the system is now internal: when it halts, one can consider objects $a \in T$, in the elementary membrane as a result (no other objects will be there). Let the superscript int stand for systems with internal output and let subscript ne mean that no rule uses the environment and the skin membrane.

Corollary 4.9.1 $PsProP^{1,int}_{2,ne}(ncoo, sym_1, anti_1) = PsRE$
$PsProP^{1,int}_{2,ne}(ncoo, sym_2) = PsRE.$

4.10 Concluding Remarks

We have studied proton pumping P systems, a variant of P systems which is both biologically motivated and mathematically elegant. Since every object only carries a finite amount of information, the cooperation of objects (i.e.,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

the exchange of information) is crucial to obtain any non-trivial computational device. Here, the cooperation is reduced to the minimum: objects can only cooperate directly with protons, by moving together to another region.

Nevertheless, this is enough to obtain computationally complete systems, even with low parameters like rewriting objects in two regions and communicating them across one membrane using just one proton.

Yet another point worth mentioning is that the constructions in the proofs have a low number of cooperative rules: four for both one-proton constructions and $|I_+| + 2|I_-| + 6$ (where I_+ is the number of ADD instructions and I_- is the number of SUB instructions in the simulated register machine) for both time-free constructions.

The one-proton results obtained here are optimal for P systems with external output in terms of number of membranes and protons, under the assumption that the skin membrane is only used to output the result: with only one membrane (i.e., output membrane) or with zero protons the behavior of the system is non-cooperative. However, there still are some challenging open problems:

- Is rewriting in both regions necessary for completeness (most of the constructions in evolution–communication P systems and proton pumping P systems heavily rely on rewriting in all regions)?
- What is the generative power of proton pumping P systems with one membrane and internal output?
- What about restricted proton pumping P systems, where the only uniport rules allowed are uniport rules of protons (i.e., protons appear in no evolution rules but in all communication rules)?
- Are four protons necessary for time-free computational completeness?

The proofs presented in this chapter are based on articles [5], [10], [7], [11] and [6]; some results are first proved in this thesis.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 5

Symport / Antiport of Small Weight

In this chapter, we first give a historical overview of the most important results obtained in the area of P systems and tissue P systems with *symport / antiport* rules, especially with respect to the development of computational completeness results improving descriptional complexity parameters as the number of membranes and cells, respectively, and the weight of the rules. Then we present the last original results.

5.1 Introduction

P systems with *symport / antiport* rules, i.e., P systems with *pure communication rules assigned to membranes*, first were introduced in [156]; symport rules move objects across a membrane together in one direction, whereas antiport rules move objects across a membrane in opposite directions. These operations are very powerful, i.e., P systems with symport / antiport rules have universal computational power with only one membrane, e.g., see [92], [101], [95].

After establishing the necessary definitions, we first give a historical overview of the most important results obtained in the area of P systems and tissue P systems with *symport / antiport* rules and review the development of computational completeness results improving descriptional complexity parameters, especially concerning the number of membranes and cells, respectively, and the weight of the rules as well as the number of objects.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

We consider two classes of P systems with minimal cooperation, i.e., P systems with symport / antiport rules of weight one and P systems with symport rules of weight two. We present the following results: P systems with minimal cooperation are computationally complete with three membranes, while tissue P systems with minimal cooperation are computationally complete with two cells, even in a deterministic way.

Moreover, P systems with minimal cooperation are computationally complete with only two membranes modulo some initial segment: In P systems with symport / antiport rules of weight one, only three superfluous objects remain in the output membrane at the end of a halting computation, whereas in P systems with symport rules of weight two six additional objects remain. For both variants, in [22] it has been shown that two membranes are enough to obtain computational completeness modulo a terminal alphabet; following [21], we now show that the use of a terminal alphabet can be avoided for the price of superfluous objects remaining in the output membrane at the end of a halting computation. So far we were not able to completely avoid these additional objects, hence, it remains as an interesting question how to reduce their number.

Then we show another construction from [21]: P systems with only one membrane and symport rules of weight three can generate any recursively enumerable language with only seven additional symbols remaining in the skin membrane at the end of a halting computation, which improves the result of [100] where thirteen superfluous symbols remained.

5.2 Definitions

We recall from the Formal Language Theory that

$$N_kRE = \{k + M \mid M \in NRE\}, \text{ where } k + M = \{k + n \mid n \in M\}.$$

5.2.1 P Systems with Symport / Antiport Rules

A *P system with symport / antiport rules* is a construct

$$\Pi = (O, E, \mu, w_1, \cdots, w_k, R_1, \cdots, R_k, i_0)$$

where:

1. O is a finite alphabet of symbols called *objects*;

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

2. μ is a *membrane structure* consisting of k membranes that are labelled in a one-to-one manner by $1, 2, \ldots, k$;

3. $w_i \in O^*$, for each $1 \leq i \leq k$, is a finite multiset of objects associated with the region i (delimited by membrane i);

4. $E \subseteq O$ is the set of objects that appear in the environment in an infinite number of copies;

5. R_i, for each $1 \leq i \leq k$, is a finite set of symport / antiport rules associated with membrane i; these rules are of the forms (x, in) and (y, out) (**symport** rules) and $(y, out; x, in)$ (**antiport** rules), respectively, where $x, y \in O^+$;

6. i_0 is the label of an elementary membrane of μ that identifies the corresponding output region.

A P system with symport / antiport rules is defined as a computational device consisting of a set of k hierarchically nested membranes that identify k distinct regions (the membrane structure μ), where to each membrane i there are assigned a multiset of objects w_i and a finite set of symport / antiport rules R_i, $1 \leq i \leq k$. A rule $(x, in) \in R_i$ permits the objects specified by x to be moved into region i from the immediately outer region. Notice that for P systems with symport rules the rules in the skin membrane of the form (x, in), where $x \in E^*$, are forbidden. A rule $(x, out) \in R_i$ permits the multiset x to be moved from region i into the outer region. A rule $(y, out; x, in)$ permits the multisets y and x, which are situated in region i and the outer region of i, respectively, to be exchanged. It is clear that a rule can be applied if and only if the multisets involved by this rule are present in the corresponding regions. The weight of a symport rule (x, in) or (x, out) is given by $|x|$, while the weight of an antiport rule $(y, out; x, in)$ is given by $max\{|x|, |y|\}$.

As usual, a computation in a P system with symport / antiport rules is obtained by applying the rules in a non-deterministic maximally parallel manner. Specifically, in this variant, a computation is restricted to moving objects through membranes, since symport / antiport rules do not allow the system to modify the objects placed inside the regions. Initially, each region i contains the corresponding finite multiset w_i, whereas the environment contains only objects from E that appear in infinitely many copies.

A computation is successful if starting from the initial configuration, the P system reaches a configuration where no rule can be applied anymore.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The result of a successful computation is a natural number that is obtained by counting all objects (only the terminal objects as it done in [22], if in addition we specify a subset of O as the set of terminal symbols) present in region i_0. Given a P system Π, the set of natural numbers computed in this way by Π is denoted by $N(\Pi)$. If the multiplicity of each (terminal) object is counted separately, then a vector of natural numbers is obtained, denoted by $Ps(\Pi)$, see [163].

By

$$NO_nP_m(sym_s, anti_t)$$

we denote the family of sets of natural numbers (non-negative integers) that are generated by a P system with symport / antiport rules having at most $n > 0$ objects in O, at least $m > 0$ membranes, symport rules of size at most $s \geq 0$, and antiport rules of size at most $t \geq 0$. By

$$N_kO_nP_m(sym_s, anti_t)$$

we denote the corresponding families of recursively enumerable sets of natural numbers without initial segment $\{0, 1, \ldots, k-1\}$. If we replace numbers by vectors, then in the notations above N is replaced by Ps. When any of the parameters m, n, s, t is not bounded, it is replaced by $*$; if the number of objects n is unbounded, we also may just omit n. If $s = 0$, then we may even omit sym_s; if $t = 0$, then we may even omit $anti_t$.

It may happen that P system with symport / antiport (symport) rules can simulate deterministic register machines (i.e., register machines where in each ADD instruction $q_1 : (A(r), q_2, q_3)$ the labels q_2 and q_3 are equal) in a deterministic way, i.e., from each configuration of the P system we can derive at most one other configuration. Then, when considering these P systems as accepting devices (the input from a set in $PsRE$ is put as an additional multiset into some specified membrane of the P system), we can get deterministic accepting P systems; the corresponding families of recursively enumerable sets of natural numbers then are denoted in the same way as before, adding the prefix D and the subscript a to N or to Ps; e.g., from the results proved in [96] and [88] we immediately obtain

$$PsRE = DPs_aOP_1(anti_2).$$

Sometimes, the results we recall use the intersection with a terminal alphabet, in that way avoiding superfluous symbols to be counted as a result

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

of a halting computation. In that case, we add the suffix $_T$ at the end of the corresponding notation.

5.2.2 Tissue P Systems with Symport / Antiport Rules

The inspiration for *tissue P systems* comes from two sides. From one hand, P systems previously introduced may be viewed as transformations of labels associated to nodes of a tree. Therefore, it is natural to consider same transformations on a graph. From the other hand, they may be obtained by following the same reflections as for P systems, but starting from a tissue of cells and no more from a single cell.

Tissue P systems were first considered by Gh. Păun and T. Yokomori in [171] and [172], and then in [140]. They have reacher possibilities and the advantages of new topology have to be investigated. Tissue P systems with symport / antiport were first considered in [163] where several results having different values of parameters (graph size, maximal size of connected component, weight of symport and antiport rules) are presented.

Tissue-like P systems with channel states were investigated in [97]. Here we deal with the following type of systems (omitting the channel states):

A tissue P system (of degree $m \geq 1$) with symport / antiport rules is a construct

$$\Pi = \left(m, O, E, w_1, \cdots, w_m, ch, \left(R_{(i,j)}\right)_{(i,j)\in ch}\right)$$

where

- m is the number of cells,
- O is the alphabet of *objects*,
- $E \subseteq O$ is the set of objects that appear in the environment in an unbounded number,
- $w_1, \ldots, w_m$ are strings over O representing the *initial* multiset of *objects* present in the cells of the system (it is assumed that the m cells are labelled with $1, 2, \ldots, m$),

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- $ch \subseteq \{(i,j) \mid i,j \in \{0,1,2,\ldots,m\}, (i,j) \neq (0,0)\}$ is the set of links (*channels*) between cells (these were called *synapses* in [97]; 0 indicates the environment), $R_{(i,j)}$ is a finite set of symport / antiport rules associated with the channel $(i,j) \in ch$.

A *symport / antiport rule* of the form y/λ, λ/x or y/x, respectively, $x, y \in O^+$, from $R_{(i,j)}$ for the ordered pair (i,j) of cells means moving the objects specified by y from cell i (from the environment, if $i = 0$) to cell j, at the same time moving the objects specified by x in the opposite direction. For short, we shall also speak of a *tissue P system* only when dealing with a *tissue P system with symport / antiport rules* as defined above.

The computation starts with the multisets specified by $w_1, \ldots, w_m$ in the m cells; in each time unit, a rule is used on each channel for which a rule can be used (if no rule is applicable for a channel, then no object passes over it). Therefore, the use of rules is sequential at the level of each channel, but it is parallel at the level of the system: all channels which can use a rule must do it (the system is synchronously evolving). The computation is successful if and only if it halts.

The result of a halting computation is the number described by the multiplicity of objects present in cell 1 (or in the first k cells) in the halting configuration. The set of all (vectors of) natural numbers computed in this way by the system Π is denoted by $N(\Pi)$ ($Ps(\Pi)$). The family of sets $N(\Pi)$ ($Ps(\Pi)$) of (vectors of) natural numbers computed as above by systems with at most $n > 0$ symbols and $m > 0$ cells as well as with symport rules of weight $s \geq 0$ and antiport rules of weight $t \geq 0$ is denoted by

$$NO_n t' P_m(sym_s, anti_t) \ (PsO_n t' P_m(sym_s, anti_t)).$$

When any of the parameters m, n, s, t is not bounded, it is replaced by $*$.

In [97], only channels (i,j) with $i \neq j$ are allowed, and, moreover, for any i, j only one channel out of $\{(i,j), (j,i)\}$ is allowed, i.e., between two cells (or one cell and the environment) only one channel is allowed (this technical detail may influence considerably the computational power). The family of sets $N(\Pi)$ ($Ps(\Pi)$) of (vectors of) natural numbers computed as above by systems with at most $n > 0$ symbols and $m > 0$ cells as well as with symport rules of weight $s \geq 0$ and antiport rules of weight $t \geq 0$ is denoted by

$$NO_n t P_m(sym_s, anti_t) \ (PsO_n t P_m(sym_s, anti_t)).$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

5.3 Descriptional Complexity: A Survey

In this section we review the development of computational completeness results with respect to descriptional complexity parameters, especially concerning the number of membranes and cells, respectively, and the weight of the rules as well as the number of objects.

5.3.1 Rules Involving More Than Two Objects

We first recall results where rules involving more than two objects are used. As it was shown in [156], two membranes are enough for getting computational completeness when rules involving at most four objects, moving up to two objects in each direction, are used, i.e.,

$$NRE = NOP_2(sym_2, anti_2).$$

Using antiport. The result stated above was independently improved in [92], [101], and [95] - one membrane is enough:

$$NRE = NOP_1(sym_1, anti_2).$$

In fact, only one symport rule is needed; this can be avoided for the price of one additional object in the output region:

$$N_1RE = N_1OP_1(anti_2).$$

It is worth mentioning that the only antiport rules used are those exchanging one object by two objects.

Using symport. The history of P systems with symport only is longer. In [141] the results

$$NRE = NOP_2(sym_5) = NOP_3(sym_4) = NOP_5(sym_3)$$

are proved, whereas in [100]

$$N_{13}RE = N_{13}OP_1(sym_3)$$

is shown; the additional symbols can be avoided if a second membrane is used:

$$NRE = NOP_2(sym_3).$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

In this chapter we present the result from [21]: we can bound the number of additional symbols by 7:

$$N_7RE = N_7OP_1(sym_3).$$

Determinism. It is known that deterministic P systems with one membrane using only antiport rules of weight at most 2 (actually, only the rules exchanging one object for two objects are needed, see [96], [55]) or using only symport rules of weight at most 3 (see [96]) can accept all sets of vectors of natural numbers (in fact, this is only proved for sets of numbers, but the extension to sets of vectors is straightforward), i.e.,

$$PsRE = DPs_aOP_1(anti_2) = DPs_aOP_1(sym_3).$$

5.3.2 Minimal Cooperation

Already in [156] it was shown that

$$NRE = NOP_5(sym_2, anti_1),$$

i.e., five membranes are already enough when only rules involving two objects are used. However, both types of rules involving two objects are used: symport rules moving up to two objects in the same direction, and antiport rules moving two objects in different directions.

Minimal cooperation by antiport. We now consider P systems where symport rules move only one object and antiport rules move only two objects across the a membrane in different directions. The first proof of the computational completeness of such P systems can be found in [45]:

$$NRE = NOP_9(sym_1, anti_1),$$

i.e., these P systems have nine membranes. This first result was improved by reducing the number of membranes to six [131], five [47], and four [99, 138], and finally in [200] it was shown that

$$N_5RE = N_5OP_3(sym_1, anti_1),$$

i.e., three membranes are sufficient to generate all recursively enumerable sets of numbers (with five additional objects in the output membrane).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

In [24], a stronger result was shown where the output membrane did not contain superfluous symbols:

$$PsRE = PsOP_3(sym_1, anti_1),$$

In [22] it was shown that even two membranes are enough to obtain computational completeness, yet only modulo a terminal alphabet:

$$PsRE = PsOP_2(sym_1, anti_1)_T,$$

In this chapter we will show the result from [21]: we can bound the number of additional symbols by 3:

$$N_3RE = N_3OP_2(sym_1, anti_1).$$

Minimal cooperation by symport. We now consider P systems moving only one or two objects by a symport rule; these systems were shown to be computationally complete with four membranes in [101]:

$$NRE = NOP_4(sym_2).$$

In [24], this result was improved down to three membranes even for vectors of natural numbers:

$$PsRE = PsOP_3(sym_2).$$

Moreover, in [22] it was also shown that even two membranes are enough to obtain computational completeness (modulo a terminal alphabet):

$$PsRE = PsOP_2(sym_2)_T$$

In this chapter we will show the result from [21]: the number of additional objects in the output region can be bound by six:

$$N_6RE = N_6OP_2(sym_2)$$

The tissue case. If we do not restrict the graph of communication to be a tree, certain advantages appear. It was shown in [203] that

$$NRE = NOtP_3(sym_1, anti_1),$$

i.e., three cells are enough when using symport / antiport rules of weight one. This result was improved in [31] to two cells, again without additional

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

objects in the output cell, and an equivalent result holds if antiport rules of weight one are replaced by symport rules of weight two:

$$PsRE = PsOtP_2(sym_1, anti_1) = PsOtP_2(sym_2).$$

Moreover, it was shown in the same article that accepting can be done deterministically:

$$PsRE = DPs_aOtP_2(sym_1, anti_1) = DPs_aOtP_2(sym_2).$$

A nice aspect of the proof is that it not only holds true for P systems with channels operating sequentially (as it is usually defined for tissue P systems), but also for P systems with channels operating in a maximally parallel way (like in standard P systems, generalizing the region communication structure of P systems to the arbitrary graph structure of tissue P systems).

Below computational completeness. In [31], it was also shown that

$$NOP_1(sym_1, anti_1) \cup NOtP_1(sym_1, anti_1) \subseteq NFIN.$$

Together with the counterpart results for symport systems,

$$NOP_1(sym_2) \cup NOtP_1(sym_2) \subseteq NFIN$$

obtained in [100], this is enough to state the optimality of the computational completeness results for the two-membrane/ two-cell systems.

The most interesting open questions remaining in the cases considered so far concern the possibility to reduce the number of extra objects in the output region in some of the results stated above.

5.4 Three Membranes

Two classes of symport / antiport P systems with three membranes and with minimal cooperation, namely P systems with symport / antiport rules of size one and P systems with symport rules of size two are computationally complete: they generate all recursively enumerable sets of vectors of non-negative integers. The result of computation is obtained in the elementary membrane.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

5.4.1 Symport / Antiport of Weight One

Theorem 5.4.1 $NOP_3(sym_1, anti_1) = NRE$.

Proof. We prove this result in the following way. We shall simulate a non-deterministic counter automaton $M = (d, Q, q_0, q_f, P)$ which starts with empty counters. We also suppose that all instructions from P are labelled in a one-to-one manner with $\{1, \cdots, n\} = I$. Denote by I_+ ($I_+ \subseteq I$) a set of labels of "increment" instructions, by I_- ($I_- \subseteq I$) a set of labels of "decrement" instructions and $I_{=0}$ ($I_{=0} \subseteq I$) is a set of labels of "zero test" instructions. We also define a set C associated to the counters: $C = \{c_k \mid 1 \leq k \leq d\}$.

We construct a P system Π with the following membrane structure:

$$[_1\ [_2\ [_3\ \ ]_3\ ]_2\ ]_1$$

The functioning of this system may be split in three stages:

1. Preparation of the system for the computation.
2. The simulation of instructions of the counter automaton.
3. Terminating the computation.

We code the counter automaton as follows. At each moment (after stage one) region 1 holds the current state of the automaton, represented by a symbol $q_i \in Q$, region 2 keeps the value of all counters, represented by the number of occurrences of symbols $c_k \in C$. We simulate the instructions of the counter automaton and we use for this simulation the symbols $c_k \in C$, $a_j, b_j, d_j, e_j,\ j \in I$. During the first stage we bring from the environment an arbitrary number of symbols b_j into region 3, symbols d_j into region 2 and symbols c_k into region 1. We suppose that we have enough symbols in the corresponding membranes to perform the computation. We also use the following idea: we bring from the environment symbols c_k into region 1 all time during the computation. This process may be stopped only if all stages finish correctly. Otherwise, the computation will never stop.

We split our proof in several parts which depend on the logical separation of the behavior of the system. We will present rules and initial symbols for each part, but we remark that the system that we present is the union of all these parts.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

We construct the P system Π as follows:

$$\begin{aligned}
\Pi &= (O, E, [_1\, [_2\, [_3\ ]_3\,]_2\,]_1, w_1, w_2, w_3, R_1, R_2, R_3, 3),\\
O &= E \cup \{f_j \mid j \in I\} \cup \{m_i \mid 1 \le i \le 5\}\\
&\cup \{l_7, l_8, g_1, g_2, g_3, I_a, I_1, I_2, I_3, I_c, O_b, O_2, i, t, \#_0, \#_1, \#_2\},\\
E &= \{a_j, b_j, d_j, e_j \mid j \in I\} \cup \{c_k \mid c_k \in C\}\\
&\cup \{q_i \mid q_i \in Q\} \cup \{l_i \mid 1 \le i \le 6\},\\
w_1 &= I_1 I_2 I_3 O_2 g_2 i l_7 l_8 \#_1 \#_2,\\
w_2 &= I_c t m_1 m_2 \#_0,\\
w_3 &= I_a O_b g_1 g_3 m_3 m_4 m_5 \prod_{j \in I} f_j,\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,f} \cup R_{i,a},\ 1 \le i \le 3.
\end{aligned}$$

The rules are given by phases: START (stage 1), RUN (stage 2), FIN (stage 3) and AUX.

AUX.

$$\begin{aligned}
R_{1,a} &= \{\mathtt{1a1} : (I_c, in), \mathtt{1a2} : (I_1, in)\} \cup \{\mathtt{1a3} : (I_c, out; c_k, in) \mid c_k \in C\}\\
&\cup \{\mathtt{1a4} : (I_1, out; b_j, in) \mid j \in I\} \cup \{\mathtt{1a5} : (I_1, out; d_j, in) \mid j \in I_{=0}\}\\
&\cup \{\mathtt{1a6} : (\#_0, in), \mathtt{1a7} : (\#_0, out)\},\\
R_{2,a} &= \{\mathtt{2a1} : (O_b, out), \mathtt{2a2} : (I_a, in), \mathtt{2a3} : (I_2, in)\}\\
&\cup \{\mathtt{2a4} : (b_j, out; O_b, in) \mid j \in I_-\} \cup \{\mathtt{2a5} : (I_a, out; a_j, in) \mid j \in I_+\}\\
&\cup \{\mathtt{2a6} : (I_2, out; b_j, in) \mid j \in I\} \cup \{\mathtt{2a7} : (I_2, out; d_j, in) \mid j \in I_{=0}\},\\
R_{3,a} &= \{\mathtt{3a1} : (O_2, out), \mathtt{3a2} : (I_3, in), \mathtt{3a3} : (I_3, out; c_1, in)\}\\
&\cup \{\mathtt{3a4} : (x, out; O_2, in) \mid x \in \{I_1, I_2, g_2, l_1, l_2, l_3, l_7\}\}\\
&\cup \{\mathtt{3a5} : (a_j, out; O_2, in) \mid j \in I\}\\
&\cup \{\mathtt{3a6} : (\#_i, in), \mathtt{3a7} : (\#_i, out) \mid 1 \le i \le 2\}.
\end{aligned}$$

Symbols I_a, I_1, I_2, I_3, I_c bring symbols inside some membrane and return. Symbols O_1, O_b take symbols outside some membrane and return. Symbols $\#_0, \#_1, \#_2$ check for "invalid" computation.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

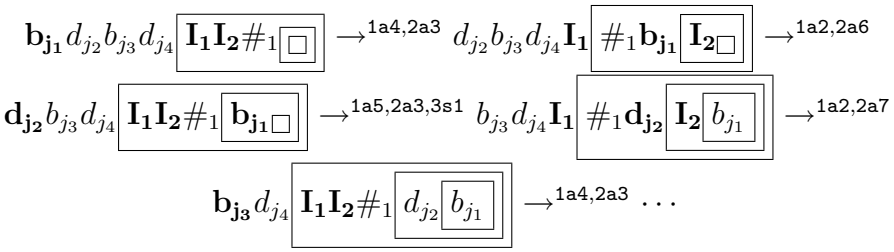


Figure 5.1: Bringing objects b_j, d_j.

START.

$$
\begin{aligned}
R_{1,s} &= \{\texttt{1s1} : (g_3, out; q_0, in)\},\\
R_{2,s} &= \{\texttt{2s1} : (I_2, out; \#_1, in), \texttt{2s2} : (t, out; I_1, in), \texttt{2s3} : (I_2, out; t, in)\}\\
&\cup \{\texttt{2s4} : (g_1, out; g_2, in), \texttt{2s5} : (I_c, out; g_1, in), \texttt{2s6} : (g_3, out; i, in)\},\\
R_{3,s} &= \{\texttt{3s1} : (b_j, in) \mid j \in I\} \cup \{\texttt{3s2} : (g_1, out; I_1, in)\}\\
&\cup \{\texttt{3s3} : (g_3, out; g_2, in), \texttt{3s4} : (I_1, out; I_2, in)\}\\
&\cup \{\texttt{3s5} : (O_b, out; I_1, in), \texttt{3s6} : (I_a, out; i, in)\}.
\end{aligned}
$$

Symbols I_1, I_2 bring from environment "sufficiently many" symbols d_j in region 2 and a "correct number of" symbols b_j in region 3 for the computation (rules `1a4,2a3,1a2,2a6,1a5,3s1,2a7`). We illustrate this process by Figure 5.1.

The figures in this paper describe different stages of evolution of the P system given in the corresponding theorem. For simplicity, we focus on explaining a particular stage and omit the objects that do not participate in the evolution at that time. Each rectangle represents a membrane, each variable represents a copy of object in a corresponding membrane (symbols outside of the rectangle are in the environment). In each step, the symbols that will evolve (will be moved) are written in boldface. The labels of the applied rules are written above the $\rightarrow$ symbol.

Notice that I_2 cannot be idle, as it immediately leads to infinite computation (rules `2s1,3a6,3a7`), so d_j and b_j in region 1 must be moved to region 2 by I_2 (rules `2a6` and `2a7`).

At some point, I_1 stops bringing symbols d_j,b_j. I_1 and I_2 are removed from their "pumping" positions, I_c is placed in region 1, where it can "pump" symbols c_k into the skin membrane, and q_0 is brought into region 1 to start

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

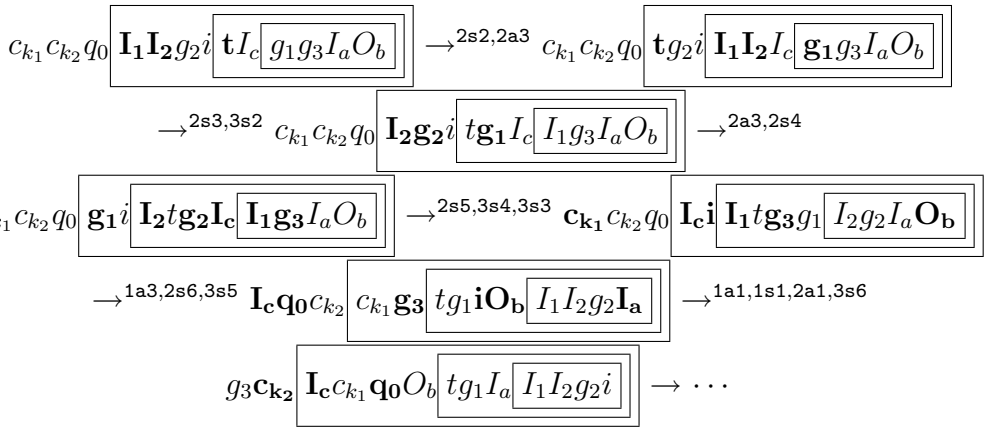


Figure 5.2: Ending of the initialization (stage 1).

the simulation of the register machine. In the meantime I_a reaches region 2 and O_b reaches region 1. Notice that both $(g_1, out; I_1, in)$ and $(O_b, out; I_1, in)$ from $R_{3,s}$ are applied, in either order (Figure 5.2).

RUN.

$$
\begin{aligned}
R_{1,r} &= \{\texttt{1r1} : (q_i, out; a_j, in), \texttt{1r2} : (b_j, out; q_l, in) \\
&\quad \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, = 0\}\} \\
&\cup \{\texttt{1r3} : (d_j, out; e_j, in) \mid (j : q_i \to q_l, k = 0) \in P\}, \\
R_{2,r} &= \{\texttt{2r1} : (b_j, out; c_k, in) \mid (j : q_i \to q_l, k+) \in P\}\} \\
&\cup \{\texttt{2r2} : (c_k, out; a_j, in) \mid (j : q_i \to q_l, k-) \in P\} \\
&\cup \{\texttt{2r3} : (d_j, out; a_j, in), \texttt{2r4} : (c_k, out; e_j, in), \\
&\quad \texttt{2r5} : (b_j, out; e_j, in) \mid (j : q_i \to q_l, k = 0) \in P\}, \\
R_{3,r} &= \{\texttt{3r1} : (b_j, out; a_j, in) \mid (j : q_i \to q_l, k+) \in P\} \\
&\cup \{\texttt{3r2} : (b_j, out; a_j, in) \mid (j : q_i \to q_l, k-) \in P\} \\
&\cup \{\texttt{3r3} : (f_j, out; a_j, in), \texttt{3r4} : (b_j, out; f_j, in) \\
&\quad \mid (j : q_i \to q_l, k = 0) \in P\}.
\end{aligned}
$$

While I_c is bringing symbols c_k into the skin membrane (rules `1a1,1a3`), instructions $(j : q_i \to q_l, k\gamma)$, $\gamma \in \{+, -, = 0\}$ of the register machine are simulated.

"Increment" instruction: see Figure 5.3.

"Decrement" instruction: see Figure 5.4.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

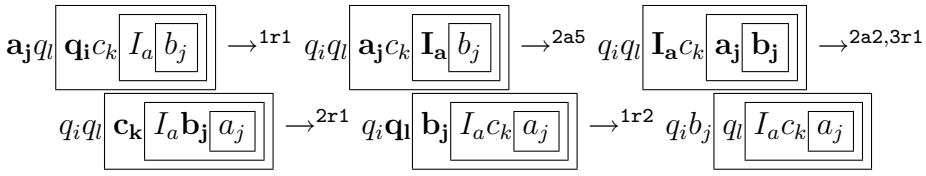


Figure 5.3: q_i replaced by q_l, c_k moved into region 2.

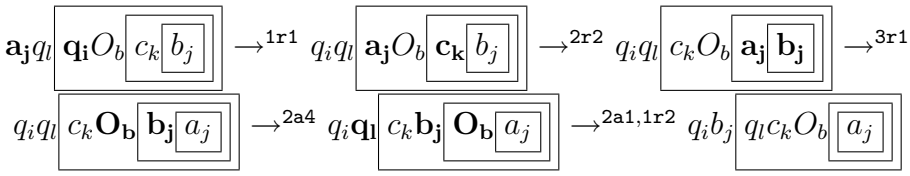


Figure 5.4: q_i replaced by q_l, c_k removed from region 2.

Checking for zero. q_i replaced by q_l if there is no c_k in region 2 (Figure 5.5), otherwise e_j exchanges with c_k and b_j remains in region 2 (Figure 5.6).

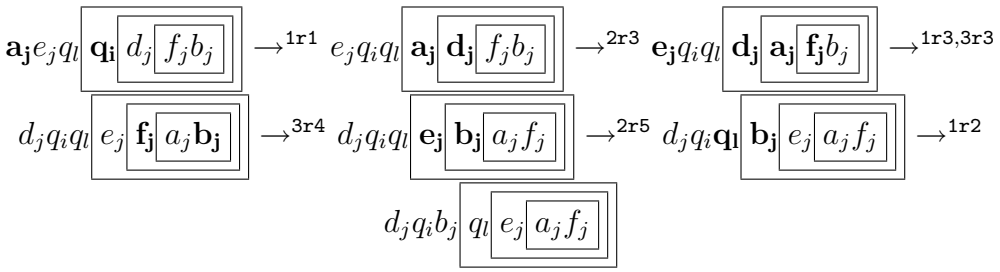


Figure 5.5: "Zero test" instruction. There is no c_k in region 2.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

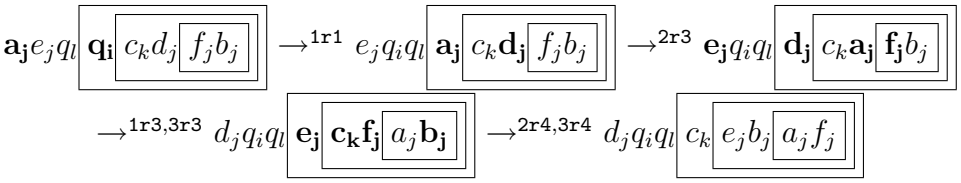


Figure 5.6: "Zero test" instruction. There is c_k in region 2.

FIN.

$$
\begin{aligned}
R_{1,f} &= \{\mathtt{1f1} : (m_1, out; l_1, in), \mathtt{1f2} : (\#_1, out; m_1, in)\} \\
&\cup \{\mathtt{1f3} : (m_2, out; l_2, in), \mathtt{1f4} : (m_3, out; l_3, in), \mathtt{1f5} : (m_4, out; l_4, in)\} \\
&\cup \{\mathtt{1f6} : (l_4, out; l_5, in), \mathtt{1f7} : (m_5, out; l_6, in)\}, \\
R_{2,f} &= \{\mathtt{2f1} : (m_1, out; q_f, in), \mathtt{2f2} : (q_f, out; l_7, in), \mathtt{2f3} : (m_2, out; l_1, in)\} \\
&\cup \{\mathtt{2f4} : (m_3, out; O_2, in), \mathtt{2f5} : (m_4, out; I_3, in), \mathtt{2f6} : (I_3, out; l_2, in)\} \\
&\cup \{\mathtt{2f7} : (m_5, out; l_8, in), \mathtt{2f8} : (l_8, out; I_c, in), \mathtt{2f9} : (c_1, out; l_6, in)\} \\
&\cup \{\mathtt{2fa} : (l_6, out; \#_2, in), \mathtt{2fb} : (l_3, in), \mathtt{2fc} : (\#_0, out, l_5, in)\} \\
&\cup \{\mathtt{2fd} : (l_3, out, l_5, in)\}, \\
R_{3,f} &= \{\mathtt{3f1} : (m_3, out; l_7, in), \mathtt{3f2} : (m_4, out; l_1, in), \mathtt{3f3} : (m_5, out; l_2, in)\} \\
&\cup \{\mathtt{3f4} : (b_j, out; l_3, in) \mid j \in I\}.
\end{aligned}
$$

If a successful computation of the register machine is correctly simulated, then q_f will appear in region 1. $\#_1$ is removed from region 1, and a chain reaction is started, during which symbols l_i move inside the membrane structure, and symbols m_i move outside the membrane structure (Figure 5.7).

Now O_2 will pump outside the elementary membrane any symbol which stays there, except c_1 (rules `3a1, 3a4, 3a5`). m_4 will exchange with I_3 (rule `2f5`), and the latter will pump symbols c_1 into the elementary membrane (rules `3a2, 3a3`), and eventually exchange with l_2 (rule `2f6`).

Object m_3 comes to the environment in exchange for l_3 (rule `1f4`), which goes to membrane 2 (rule `2fb`), and stays there if there is no object b_j in the elementary membrane (otherwise l_3 will exchange with b_j by rule `3f4`). Object m_4 comes to the environment in exchange for l_4 (rule `1f5`), which brings l_5 in the skin. l_5 then exchanges with l_3 by rule `2fd`. Notice that presence of b_j in region 3 will force l_5 to move $\#_0$ in region 1 (rule `2fc`),

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$l_1 l_1 l_2 \boxed{\mathbf{q_f} l_7 O_2 I_3 \#_1 \boxed{\mathbf{m_1} m_2 \boxed{m_3 m_4}}} \rightarrow^{\texttt{2f1}} \mathbf{l_1} l_1 l_2 \boxed{\mathbf{m_1 l_7} O_2 I_3 \#_1 \boxed{\mathbf{q_f} m_2 \boxed{m_3 m_4}}}$$
$$\rightarrow^{\texttt{1f1,2f2}} \mathbf{m_1} l_1 l_2 \boxed{\mathbf{l_1} q_f O_2 I_3 \#_{\mathbf{1}} \boxed{\mathbf{l_7 m_2} \boxed{\mathbf{m_3} m_4}}} \rightarrow^{\texttt{1f2,2f3,3f1}}$$
$$\#_1 \mathbf{l_1 l_2} \boxed{\mathbf{m_2} q_f \mathbf{O_2 m_1} I_3 \boxed{\mathbf{m_3 l_1} \boxed{l_7 \mathbf{m_4}}}} \rightarrow^{\texttt{1f1,1f3,2f4,3f2}}$$
$$\#_1 m_1 m_2 \boxed{l_2 q_f m_3 l_1 I_3 \boxed{O_2 m_4 \boxed{l_7 l_1}}}.$$

Figure 5.7: Beginning of the termination (stage 3).

leading to an infinite computation (rules `1a6, 1a7`), as l_3 will be situated in region 3.

Finally (after I_3 returns to region 1 and l_2 comes in region 2 by rule `2f6`), l_2 moves m_5 into region 2 (rule `3f3`), and the latter exchanges with l_8 (rule `2f7`) and then with l_6 (rule `1f7`). At some point l_8 moves I_c into region 2 (rule `2f8`), to finish pumping objects c_k. As for l_6 in membrane 1, it guarantees that no more objects c_1 remain in membrane 2 (otherwise it moves $\#_2$ in membrane 2 (rules `2f9, 2fa`), leading to an infinite computation (rule `3a6, 3a7`)).

If the computation halts, then the elementary membrane will only contain objects c_1, in the multiplicity of the value of the first register of the register machine. Conversely, any computation of the register machine allows a correct simulation (from the construction). Thus, the class of P systems with symport and antiport of weight 1 generate exactly all recursively enumerable sets of non-negative integers. □

5.4.2 Symport of Weight Two

A "dual" class of systems $OP(sym_1, anti_1)$ is the class $OP(sym_2)$ where two objects are moved across the membrane in the same direction rather than in the opposite ones. We now prove a similar result for the other class.

Theorem 5.4.2 $NOP_3(sym_2) = NRE$.

Proof. As in the proof of Theorem 1 we simulate a non-deterministic counter automaton $M = (d, Q, q_0, q_f, P)$ which starts with empty counters.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Again we suppose that all instructions from P are labelled in a one-to-one manner with $\{1,\ldots,n\} = I$, and I_+ ($I_+ \subseteq I$) is a set of labels of "increment" instructions, I_- ($I_- \subseteq I$) is a set of labels of "decrement" instructions, and $I_{=0}$ ($I_{=0} \subseteq I$) is a set of labels of "zero test" instructions. A set C is associated to the counters: $C = \{c_k \mid 1 \leq k \leq d\}$.

We construct the P system Π_2 as follows:

$$\begin{aligned}
\Pi_2 &= (O, E, [_1\, [_2\, [_3\ ]_3\,]_2\,]_1, w_1, w_2, w_3, R_1, R_2, R_3, 3),\\
O &= E \cup \{d_j, e_j \mid j \in I\} \cup \{t_i \mid 0 \leq i \leq 10\}\\
&\cup\ \{g_1, g_3, I_a, I_1, I_2, I_c, O_b, m_1, m_2, s_1, s_2, l_1, l_2, \#_1, \#_2\}\\
&\cup\ \{q_i \mid q_i \in Q\},\\
E &= \{a_j, b_j \mid j \in I\} \cup \{c_k \mid c_k \in C\} \cup \{l_i \mid 3 \leq i \leq 8\} \cup \{g_2\},\\
w_1 &= t_0 t_1 t_2 t_3 t_4 I_1 I_2 I_a l_1 l_2 \#_1 \prod_{j\in I} e_j \prod_{q_i \in Q} q_i,\\
w_2 &= t_5 t_6 t_7 t_8 t_9 t_{10} I_c g_3 s_1 m_2 \#_2 \prod_{j\in I} d_j,\\
w_3 &= g_1 O_b s_2 m_1,\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,m} \cup R_{i,c} \cup R_{i,f} \cup R_{i,a},\ 1 \leq i \leq 3.
\end{aligned}$$

The functioning of this system may be split in three stages like it is done in Theorem 5.4.1.

We code the counter automaton as follows. At each moment (after stage one) the environment holds the current state of the automaton, represented by a symbol $q_i \in Q$, the membrane 2 holds the value of all counters, represented by the number of occurrences of symbols $c_k \in C$. We simulate the instructions of the counter automaton and we use for this simulation the symbols $c_k \in C$, a_j, b_j, d_j, e_j, $j \in I$. During the first stage we bring from environment in the membrane 3 an arbitrary number of symbols b_j. We suppose that we have enough symbols b_j in membrane 3 to perform the computation. We also use the following idea: we bring from environment to membrane 1 the symbols c_k all time during the computation. This process may be stopped only if all stages completed correctly. Otherwise, the computation will never stop.

We split our proof in several parts which depend on the logical separation of the behavior of the system. We will present rules and initial symbols for each part, but we remark that the system that we present is the union of all these parts.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The rules R_i are given by phases: START (stage 1); RUN (stage 2); MOVE, CLEANUP and FIN (stage 3), and AUX.

AUX.

$$\begin{aligned}
R_{1,a} &= \{\texttt{1a1} : (I_c, out), \texttt{1a2} : (I_1, out)\} \cup \{\texttt{1a3} : (I_c c_k, in) \mid c_k \in C\} \\
&\cup \{\texttt{1a4} : (I_1 b_j, in) \mid j \in I\} \cup \{\texttt{1a5} : (\#_2, in), \texttt{1a6} : (\#_2, out)\}, \\
R_{2,a} &= \{\texttt{2a1} : (O_b, in), \texttt{2a2} : (I_a, out), \texttt{2a3} : (I_2, out)\} \\
&\cup \{\texttt{2a4} : (O_b b_j, out) \mid j \in I_+\} \cup \{\texttt{2a5} : (I_a a_j, in) \mid j \in I_-\} \\
&\cup \{\texttt{2a6} : (I_2 b_j, in) \mid j \in I\}, \\
R_{3,a} &= \{\texttt{3a1} : (\#_1, in), \texttt{3a2} : (\#_1, out)\} \\
&\cup \{\texttt{3a3} : (s_i, in), \texttt{3a4} : (s_i, out) \mid 1 \leq i \leq 2\}.
\end{aligned}$$

Symbol I_1 brings symbols b_j inside membrane 1 and returns to the environment. Symbol I_c brings symbols c_k inside membrane 1 and returns to the environment. Symbol I_2 brings symbols b_j inside membrane 2 and returns to membrane 1. Symbol I_a brings symbols a_j inside membrane 2 and returns to membrane 1. Symbol O_b takes symbols b_j outside membrane 2 and returns. Symbols $\#_1, \#_2$ check for "invalid" computations. Symbols s_1, s_2 remember whether the derivation step is even or odd.

START.

$$\begin{aligned}
R_{1,s} &= \{\texttt{1s1} : (g_1 t_2, out), \texttt{1s2} : (t_2 g_2, in), \texttt{1s3} : (g_3 q_0, out)\}, \\
R_{2,s} &= \{\texttt{2s1} : (t_0 I_2, in), \texttt{2s2} : (I_2 \#_1, in), \texttt{2s3} : (I_1 t_1, in)\} \\
&\cup \{\texttt{2s4} : (g_1 I_c, out), \texttt{2s5} : (g_2 t_3, in), \texttt{2s6} : (t_3 g_3, out)\}, \\
R_{3,s} &= \{\texttt{3s1} : (b_j, in) \mid j \in I\} \cup \{\texttt{3s2} : (I_2 t_1, in), \texttt{3s3} : (I_1 t_7, in)\} \\
&\cup \{\texttt{3s4} : (t_7 g_1, out), \texttt{3s5} : (g_2 t_{10}, in), \texttt{3s6} : (t_{10} O_b, out)\}.
\end{aligned}$$

Symbols I_1, I_2 bring from environment a "correct number of" symbols b_j in region 3 for the computation (rules `1a2, 1a4, 2a6, 2a3, 3s1`) (see Figure 5.8). Notice that I_2 cannot be idle, as it immediately leads to infinite computation (rules `2s2, 3a1, 3a2`), so b_j in region 1 must be moved by I_2 by rule `2a6`.

At some point, I_1 stops bringing symbols b_j. I_1 and I_2 are removed from their "pumping" positions, I_c is placed in region 1, where it can "pump" symbols c_k into the skin membrane, and q_0 is brought into the environment to start the simulation of the register machine. In the meantime O_b reaches region 2 (Figure 5.9).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

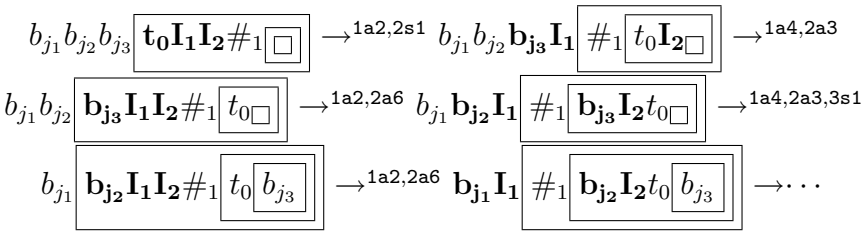


Figure 5.8: Bringing objects b_j.

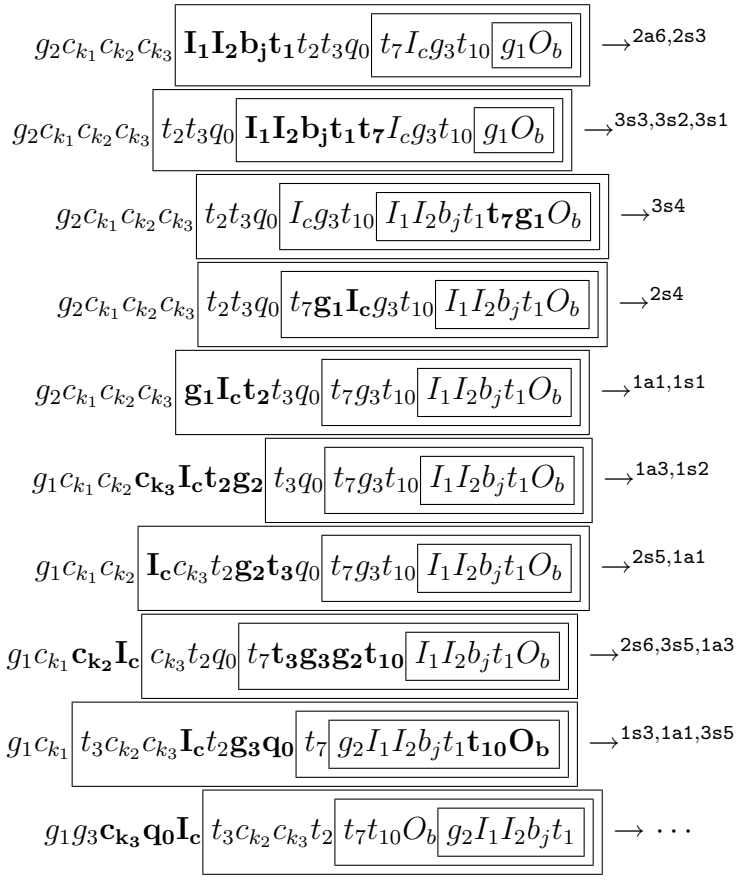


Figure 5.9: End of the initialization (stage 1).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

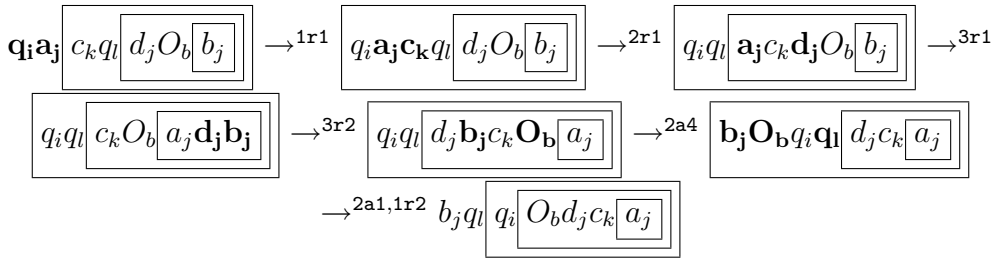


Figure 5.10: q_i replaced by q_l, c_k moved into region 2.

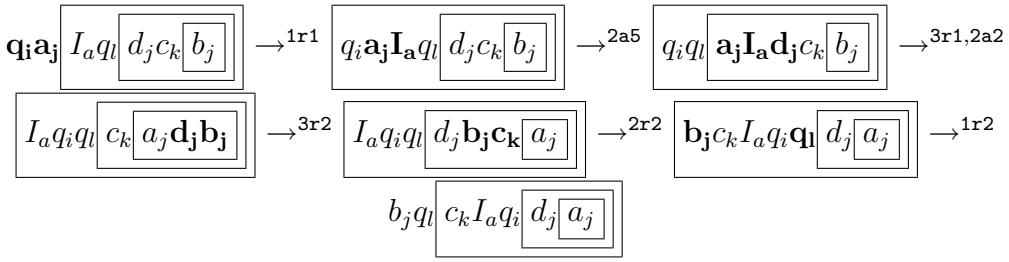


Figure 5.11: q_i replaced by q_l, c_k removed from region 2.

RUN.

$$
\begin{aligned}
R_{1,r} &= \{\texttt{1r1} : (q_i a_j, in), \texttt{1r2} : (b_j q_l, out) \\
&\quad \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, = 0\}\} \\
R_{2,r} &= \{\texttt{2r1} : (a_j c_k, in) \mid (j : q_i \to q_l, k+) \in P\} \\
&\cup \{\texttt{2r2} : (b_j c_k, out) \mid (j : q_i \to q_l, k-) \in P\} \\
&\cup \{\texttt{2r3} : (a_j e_j, in), \texttt{2r4} : (e_j c_k, out), \\
&\quad \texttt{2r5} : (e_j b_j, out) \mid (j : q_i \to q_l, k = 0) \in P\}, \\
R_{3,r} &= \{\texttt{3r1} : (a_j d_j, in), \texttt{3r2} : (d_j b_j, out) \\
&\quad \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, = 0\}\}.
\end{aligned}
$$

While I_c is bringing symbols c_k into the skin membrane (rules `1a1, 1a3`), instructions $(j : q_i \to q_l, k\gamma)$, $\gamma \in \{+, -, = 0\}$ of the register machine are simulated.

"Increment" instruction: see Figure 5.10.

"Decrement" instruction: see Figure 5.11.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

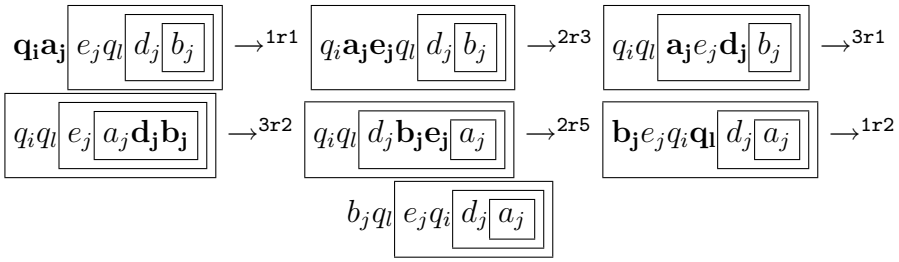


Figure 5.12: "Zero test" instruction. There is no c_k in region 2.

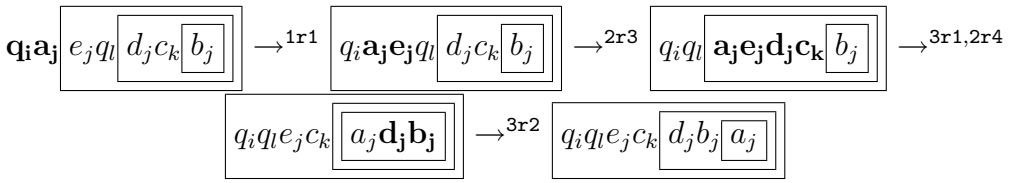


Figure 5.13: "Zero test" instruction. There is c_k in region 2.

Checking for zero. q_i replaced by q_l if there is no c_k in region 2 (Figure 5.12), otherwise e_j comes in region 1 with c_k and b_j remains in region 2 (Figure 5.13).

MOVE.

$$\begin{aligned}
R_{1,m} &= \{\texttt{1m1} : (q_f l_3, in), \texttt{1m2} : (m_1 t_4, out), \texttt{1m3} : (t_4 l_4, in)\},\\
R_{2,m} &= \{\texttt{2m1} : (l_3 l_1, in), \texttt{2m2} : (m_1 t_6, out), \texttt{2m3} : (t_6 l_2, in)\}\\
&\cup \{\texttt{2m4} : (l_2 \#_2, out)\},\\
R_{3,m} &= \{\texttt{3m1} : (l_1 c_1, in), \texttt{3m2} : (l_1, out), \texttt{3m3} : (l_3 t_5, in)\}\\
&\cup \{\texttt{3m4} : (t_5 m_1, out), \texttt{3m5} : (l_2 t_8, in)\} \cup \{\texttt{3m6} : (l_2 b_j, out) \mid j \in I\}.
\end{aligned}$$

If a successful computation of the register machine is correctly simulated, then q_f will appear in region 1. A chain reaction is started, during which symbols l_i move inside the membrane structure, and symbols m_i move outside the membrane structure. Notice that q_f brings l_3 into region 1 (rule `1m1`), then l_3 brings l_1 into region 2 (`rule 2m1`), then l_1 moves objects c_1 from region 2 into region 3 by rules `3m1` and `3m2`. Also, the system verifies that no objects b_j are present in the inner region (otherwise l_2 would bring

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

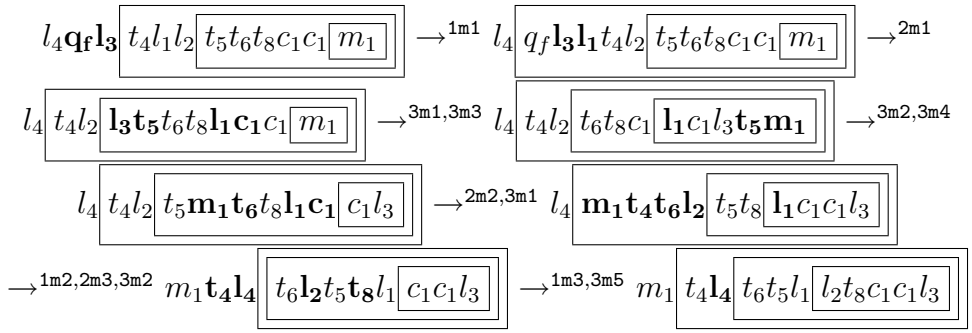


Figure 5.14: Beginning of the termination (stage 3).

$\#_2$ in region 1 (rules `3m6, 2m4`) and it immediately leads to infinite computation (rules `1a5,1a6`)) and moves l_4 into the skin membrane, as shown below (Figure 5.14).

CLEANUP.

$$\begin{aligned}
R_{1,c} &= \{\texttt{1c1} : (l_4s_1, out), \texttt{1c2} : (s_1l_5, in), \texttt{1c3} : (m_2\#_1, out)\} \\
&\cup \{\texttt{1c4} : (l_5s_2, out), \texttt{1c5} : (s_2l_7, in), \texttt{1c6} : (l_6s_2, in)\}, \\
R_{2,c} &= \{\texttt{2c1} : (l_4, in), \texttt{2c2} : (l_4s_1, out), \texttt{2c3} : (l_5t_9, in)\} \\
&\cup \{\texttt{2c4} : (t_9m_2, out), \texttt{2c5} : (l_5s_2, out)\}, \\
&\cup \{\texttt{2c6} : (l_4x, out) \mid x \in \{t_5, t_7, t_{10}\} \cup \{d_j \mid j \in I\}\},
\end{aligned}$$

$$\begin{aligned}
R_{3,c} &= \{\texttt{3c1} : (l_5, in), \texttt{3c2} : (l_5s_2, out)\} \\
&\cup \{\texttt{3c3} : (l_5x, out) \mid x \in \{I_1, I_2, g_2, t_8, l_3\} \cup \{a_j \mid j \in I\}\}.
\end{aligned}$$

Objects d_j, $j \in I$ and t_5, t_7, t_{10} are removed from region 2, and then objects a_j, $j \in I$ and I_1, I_2, g_2, t_8, l_3 are removed from the inner region. Notice that l_4 only "meets" s_1 (and l_5 only "meets" s_2) after the corresponding cleanup is completed. Really, it is easily to see that object l_4 will be in region 2 after odd steps of computation. Symbol s_1 after odd steps of computation will be located in region 3 (rules `3a3,3a4`). Thus we cannot apply rule `2c2` and can apply rule `2c6` only, until all symbols t_5, t_7, t_{10} and d_j, $j \in I$ will be removed to region 1. After that symbol l_4 waits one step and together with symbol s_1 moves to region 1 and finally to the environment (rules `2c2` and `1c1`).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

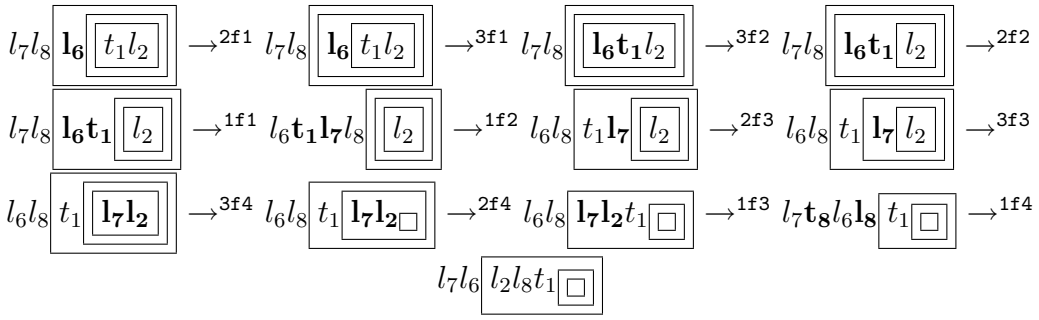


Figure 5.15: End of the termination.

So l_4 will be in the environment after even steps of computation and object l_5 will appear in region 3 after odd steps of computation (rules `1c2`, `2c3` and `3c1`). Notice that symbol s_2 can appear in region 3 after even steps of computation (rules `3a4,3a3`). Thus we cannot apply rule `3c2` and can apply rule `3c3` only, until all symbols I_1, I_2, g_2, t_8, l_3 and $a_j,\ j \in I$ will be removed to region 2. After that object l_5 moves to the environment together with symbol s_2 (rules `3c2,2c5,1c4`) and object l_6 is brought in region 1 (rule `1c6`). At that moment in membrane 3 among symbols c_1 there are only two "undesirable" symbols: t_1 and l_2.

FIN.

$$\begin{aligned}
R_{1,f} &= \{\texttt{1f1} : (l_6t_1, out), \texttt{1f2} : (t_1l_7, in), \texttt{1f3} : (l_7l_2, out), \texttt{1f4} : (l_2l_8, in)\},\\
R_{2,f} &= \{\texttt{2f1} : (l_6, in), \texttt{2f2} : (l_6t_1, out), \texttt{2f3} : (l_7, in)\}\\
&\cup\ \{\texttt{2f4} : (l_7l_2, out), \texttt{2f5} : (l_8I_c, in)\},\\
R_{3,f} &= \{\texttt{3f1} : (l_6, in), \texttt{3f2} : (l_6t_1, out), \texttt{3f3} : (l_7, in), \texttt{3f4} : (l_7l_2, out)\}.
\end{aligned}$$

Objects t_1 and l_2 are removed from the inner region, as shown below (Figure 5.15), and then l_8 moves I_c from region 1 into region 2 (rule `2f5`) so that the computation can halt.

If the computation halts, then the elementary membrane will only contain objects c_1, in the multiplicity of the value of the first register of the register machine. Conversely, any computation of the register machine allows a correct simulation (from the construction). Thus, the class of P systems with symport of weight 2 generates exactly all recursively enumerable sets of non-negative integers. □

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Final Remarks Both constructions can be easily modified to show $PsOP_3(sym_1, anti_1) = PsRE$ and $PsOP_3(sym_2) = PsRE$ by moving all output symbols c_k to the elementary membrane, as it is done for symbol c_1. In the proof of Theorem 5.4.1 we simply change rule `3a3`: $(I_3, out; c_1, in)$ by rules `3a3`: $(I_3, out; c_k, in)$ for all $c_k \in C$ and in the proof of Theorem 5.4.2 change rule `3m1`: $(l_1 c_1, in)$ by rules `3m1`: $(l_1 c_k, in)$ for all $c_k \in C$.

The questions what is the *exact* characterizations of families of numbers computed by minimal symport / antiport (symport) P systems rules with one and two membranes is still open.

Program check P systems in both theorems were tested by Vladimir Rogozhin using a modification of the simulator described in Section 3.4.

5.5 Two Cells

We consider tissue P systems with symport / antiport with minimal cooperation, i.e., when only two objects may interact. We show that two cells are enough in order to generate all recursively enumerable sets of numbers. Moreover, constructed systems simulate register machines and have purely deterministic behavior.

In this section, we consider both variants (symport / antiport of weight one and symport of weight at most two) and we show that in both cases we can construct systems defined on a graph with three nodes, i.e., two cells, that simulate any (non-)deterministic register machine. Moreover, in the deterministic case, obtained systems are also deterministic and only one evolution is possible at any time. Therefore, if the computation stops, then we are sure that the corresponding register machine stops on the provided input. Moreover, we use a very small amount of symbols present in an infinite number of copies in the environment.

5.5.1 Symport / Antiport of Weight One

Lemma 5.5.1 *For any deterministic register machine M and for any input I_n there is a tissue P system with symport / antiport of degree 2 having symport and antiport rules of weight* 1*, which simulates M on this input and produces the same result.*

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

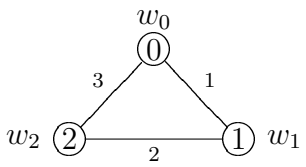
Proof. We consider an arbitrary deterministic register machine $M = (k, Q, q_0, q_f)$ and we construct a tissue P system with symport / antiport that will simulate this machine on the input I_n. We consider a more general problem: we shall simulate M on any initial configuration $(q_0, N_1, \cdots, N_k)$.

We assume, by commodity, that we may have objects initially present in a finite number of copies in the environment and we denote this multiset by w_0. We shall show later that this assumption is not necessary.

We define the system as follows.

$$\begin{aligned}
\Pi &= (2, O, E, w_0, w_1, w_2, \{(1,0),(1,2),(2,0)\}, R_{(1,0)}, R_{(1,2)}, R_{(2,0)}, 2),\\
O &= Q \cup R \cup \{A_{pq}^{+} \mid p : (A(A), q) \in Q\}\\
&\cup \{p', p'', Q_{pqs}^{-}, Q_{pqs}^{0} \mid p : (S(A), q, s) \in Q\},\\
E &= R = \{A \mid 1 \leq A \leq k\}.
\end{aligned}$$

We consider the following underlying graph G:



Below we give in tables rules and objects of our system. In fact, each w_i, $0 \leq i \leq 2$ as well as R is the union of corresponding cells in all tables.

Rule numbers follow the following convention: the second number is the number of the channel where the rule is located, the first number indicates which instruction is simulated using this rule (1 for incrementing, 2 for decrementing, 3 for stop and 0 for common parts) and the third is the number in group.

Encoding of initial configuration of M $(1 \leq j \leq k)$:

Region	Object(s)
2.	$r_j^{N_j}, q_0$

Common rules and objects ($q \in Q$, $A \in R$, $p \in Q - \{q_0\}$):

Region	Object(s)
0.	A^{∞}
1.	p

Channel	Rules
$(2,0)$	$0.3.1 : \lambda/q$

For any rule $p : (A(A), q) \in Q$ we have following rules and objects:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Region	Object(s)
1.	A^+_{pq}

Channel	Rules
$(1,0)$	$1.1.1 : q/A^+_{pq}$
$(1,2)$	$1.2.1 : A^+_{pq}/p$
$(2,0)$	$1.3.1 : A^+_{pq}/A$

For any rule $p : (S(A), q, s) \in Q$ we have following rules and objects:

Region	Object(s)
0.	p''
1.	p', Q'_{pqs}, Q^0_{pqs}

Channel	*Rules*	
$(1,0)$	$2.1.1 : Q'_{pqs}/p'$	$2.1.2 : p''/\lambda$
	$2.1.3 : Q^0_{pqs}/Q'_{pqs}$	$2.1.4 : s/Q^0_{pqs}$
$(1,2)$	$2.2.1 : p'/p$	$2.2.2 : Q^0_{pqs}/p''$
	$2.2.3 : q/Q'_{pqs}$	$2.2.4 : \lambda/Q^0_{pqs}$
$(2,0)$	$2.3.1 : p'/p''$	$2.3.2 : A/Q'_{pqs}$

Rules and objects associated to the STOP instruction:

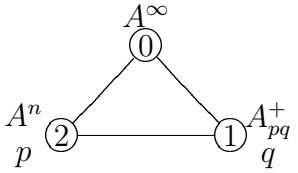
Channel	*Rules*
$(1,2)$	$3.2.1 : \lambda/q_f$

We organize system Π as follows. Region 2 contains the current configuration of machine M. Region 1 contains one copy of objects that correspond to each state. In the same region, there are additional symbols used for the simulation of the decrementing operation and which are present in one copy. Region 0 (the environment) contains symbols r_j, $1 \leq j \leq k$ which are used to increment registers. These symbols are present in an infinite number of copies.

Each configuration $(p, n_1, \ldots, n_k)$ of machine M is encoded as follows. Cell 2 contains objects r_j of the multiplicity n_j, $1 \leq j \leq k$ as well as the object p. It is easy to observe that the initial configuration of Π corresponds to an encoding of the initial configuration of M.

Now we shall discuss the simulation of instructions of M.

Incrementing Suppose M is in configuration $(p, n_1, \ldots, n_k)$ and that there is a rule $\mathbf{p} : (\mathbf{A}(\mathbf{A}), \mathbf{q})$ in P $(A = r_j)$. Suppose that the value of A is n $(n_j = n)$. This corresponds to the following configuration of Π (see below) where we indicate only symbols that we effectively use.

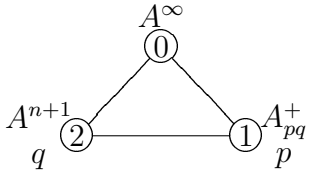


UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Now we present the evolution of the system in this case.

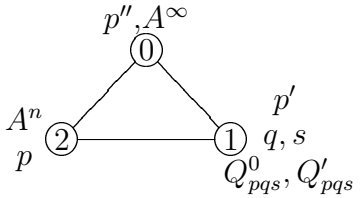
The symbol p in the second region that encodes the current state of M triggers the application of rule 1.2.1 and it exchanges with A^+_{pq} which comes to the second region.

After that, the last symbol brings an object A to region 2, which corresponds to an incrementing of register A. Further, symbol A^+_{pq} goes to region 1 and brings from there to region 0 the new state q. After that, symbol q moves to region 2. This configuration is shown below.



The last configuration differs from the first one by the following. In region 2, there is one more copy of object A and the object p was replaced by the object q. All other symbols remained on their places and the symbol p was moved to region 1 which contains as before one copy of each state of the register machine that is not current. This corresponds to the following configuration of M: $(q, n_1, \ldots, n_j + 1, \ldots, n_k)$, i.e.,, we have simulated the corresponding instruction of M.

Decrementing Suppose M is in configuration $(p, n_1, \ldots, n_k)$ and that there is a rule $\mathbf{p} : (\mathbf{S}(\mathbf{A}), \mathbf{q}, \mathbf{s})$ in P $(A = r_j)$. Suppose that the value of A is n $(n_j = n)$. This corresponds to the following configuration of Π (see below) where we indicate only symbols that we effectively use.

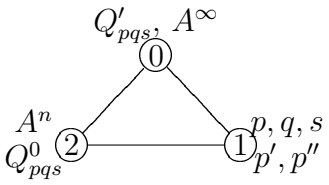


The idea of the decrementing is very simple. First we duplicate the signal that the current state is p (p' and p''). After that, these signals are propagated and if they may be synchronized, then this means that the value of the corresponding register is zero. In the other case, the corresponding register is decremented and the synchronization is no more possible.

Now we shall give more details on the evolution of the system. The symbol p in the second region that encodes the current state of M triggers the application of rule 2.2.1 and it exchanges with p' which comes to the

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

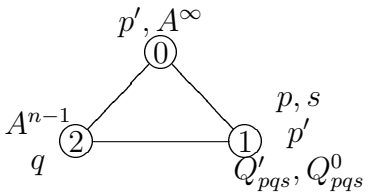
second region. After that, the last symbol goes to region 0 bringing at the same time the symbol p'' in region 2. During next step, symbol p' exchanges with Q'_{pqs} and symbol p'' exchanges with Q^0_{pqs}. The obtained configuration is shown below.



Now there are two cases, $n > 0$ and $n = 0$, and the system behaves differently in each case.

CASE A: $n > 0$

First, suppose that $n > 0$. In this case, Q'_{pqs} goes to region 2 and brings a symbol A to region 0, hence decrementing the register. At the same time symbol p'' returns to the environment. Finally, the symbol Q'_{pqs} brings the symbol q to region 2 (see configuration below).

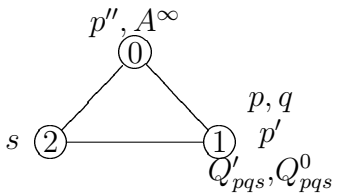


We can see that the obtained configuration differs from the first one by the following. In region 2, there is one copy of object A less and the object p was replaced by the object q. All other symbols remained in their places and the symbol p was moved to region 1 which contains as before one copy of each state of the register machine that is not current. This corresponds to the following configuration of M: $(q, n_1, \ldots, n_j - 1, \ldots, n_k)$, i.e.,, we have simulated the corresponding instruction of M.

CASE B: $n = 0$

Now suppose that $n = 0$. In this case, Q'_{pqs} remains in region 0 for one more step. After that, it exchanges with Q^0_{pqs} which returns to region 1. After that Q^0_{pqs} brings in region 0 the symbol s which moves after that to region 2 (see configuration below). We remark that the first exchange takes place only if the value of register A is equal to zero, otherwise rule 2.3.2 is applied and the exchange above cannot happen.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007



We can see that the obtained configuration differs from the first one by the following. In region 2, the object p was replaced by the object s. All other symbols remained on their places and the symbol p was moved to region 1 which contains as before one copy of each state of the register machine that is not current. This corresponds to the following configuration of M: $(s, n_1, \ldots, n_{j-1}, 0, n_{j+1}, \ldots, n_k)$, i.e., we have simulated the corresponding instruction of M.

Stop If the system is in halting state q_f, then rule 3.2.1 moves the symbol q_f to region 1. Since region 2 contain only symbols corresponding to the value of the output register, the system cannot evolve any more and the computation stops.

Remarks It is clear that we simulate the behaviour of M. Indeed, we simulate an instruction of M and all additional symbols return to their places what permits to simulate the next instruction of M. Moreover, this permits to reconstruct easily a computation in M from a successful computation in Π. For this it is enough to look for configurations which have a state symbol p in region 2. We stop the computation when rule 3.2.1 is used and symbol q_f goes to region 1. In this case, region 2 contains the result of the computation.

We remark that the assumption that $w_0 = p''$ for all decrementing instructions $p : (S(A), q, s)$ is not necessary. Indeed, we may initially place p'' in region 1. □

Theorem 5.5.1 $NOtP_2(sym_1, anti_1) = NRE$.

Proof. It is easy to observe that the system of previous lemma may simulate a non-deterministic register machine. Indeed, in order to simulate a rule $p : (A(A), q, s)$ of such machine, we use the same rules and objects as for rule $p : (A(A), q)$. We only need to add a rule $1.1.1' : s/A^+_{pq}$ to the channel $(1, 0)$. □

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

5.5.2 Symport of Weight Two

Lemma 5.5.2 *For any deterministic register machine M and for any input I_n there is a tissue P system with symport / antiport of degree 2 having only symport rules of weight at most 2, which simulates M on this input and produces the same result.*

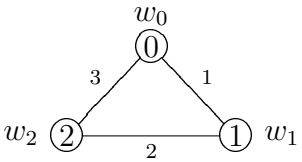
Proof. As in previous case we consider an arbitrary deterministic register machine $M = (k, Q, q_0, q_f)$ and we construct a tissue P system with symport / antiport that will simulate this machine on any initial configuration $(q_0, N_1, \ldots, N_k)$.

We assume, by commodity, that we may have objects initially present in a finite number of copies in the environment and we denote this multiset by w_0. We shall show later that this assumption is not necessary. Let m be the number of instructions of M and m_1 the number of incrementing instructions. Let $n = m + 5(m - m_1 - 1) + m_1 = 6m - 4m_1 - 5$.

We define the system as follows.

$$\begin{aligned}
\Pi &= (2, O, E, w_0, w_1, w_2, \{(1,0),(1,2),(2,0)\}, R_{(1,0)}, R_{(1,2)}, R_{(2,0)}, 2),\\
O &= Q \cup \{X_q \mid q \in Q\} \cup \{A^+_{pq} \mid p : (A(A), q) \in Q\} \cup \{Y, E, E_Y, E', S, V\}\\
&\cup \{p', p'', p^0, Z_{pqs}, Z'_{pqs}, D_{pqs}, D'_{pqs}, X_{pqs}, X_{p^0} \mid p : (S(A), q, s) \in Q\}\\
&\cup \{E_j, E'_j \mid 1 \le j \le n\} \cup R,\\
E &= R = \{A \mid 1 \le A \le k\}.
\end{aligned}$$

We consider the following underlying graph G:



Below we give in tables rules and objects of our system. In fact, each w_i, $0 \le i \le 2$, as well as R is the union of corresponding cells in all tables.

Rule numbers follow the following convention: the second number is the number of the channel where the rule is located, the first number indicates which instruction is simulated using this rule (1 for incrementing, 2 for decrementing, 3 for stop and 0 for common part) and the third is the number in group.

Encoding of initial configuration of M $(1 \le j \le k)$:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Region	Object(s)
2.	$r_j^{N_j}, q_0$

Common rules and objects ($q \in Q$, $A \in R$, $p \in Q - \{q_0\}$):

Region	Object(s)
0.	A^∞, Y
1.	p

Channel	*Rules*	
$(1,0)$	$0.1.1 : A/\lambda$	
$(2,0)$	$0.3.1 : \lambda/qY$	$0.3.2 : Y/\lambda$

For any rule $p : (A(A), q) \in Q$ we have following rules and objects:

Region	Object(s)
2.	A^+_{pq}

Channel	Rules
$(1,0)$	$1.1.1 : \lambda/A^+_{pq}q$
$(1,2)$	$1.2.1 : \lambda/pA^+_{pq}$
$(2,0)$	$1.3.1 : A^+_{pq}A/\lambda$

For any rule $p : (S(A), q, s) \in Q$ we have following rules and objects:

Region	Object(s)
0.	$p'', D_{pqs}, D''_{pqs}, Z'_{pqs}$
1.	p^0, D'_{pqs}, Z_{pqs}
2.	p'

Channel	*Rules*	
$(1,0)$	$2.1.1 : p'p^0/lambda$	$2.1.2 : \lambda/p''p^0$
	$2.1.3 : D_{pqs}D'_{pqs}/\lambda$	$2.1.4 : D''_{pqs}q/\lambda$
	$2.1.5 : \lambda/Z_{pqs}Z'_{pqs}$	$2.1.6 : Z'_{pqs}s/\lambda$
$(1,2)$	$2.2.1 : \lambda/pp'$	$2.2.2 : \lambda/D_{pqs}A$
	$2.2.3 : p''Z_{pqs}\lambda$	$2.2.4 : \lambda/D'_{pqs}Z_{pqs}$
	$2.2.5 : \lambda/D''_{pqs}V$	$2.2.6 : V\lambda$
$(2,0)$	$2.3.1 : \lambda/p'D_{pqs}$	$2.3.2 : \lambda/D'_{pqs}D''_{pqs}$
	$2.3.3 : D_{pqs}Z_{pqs}/\lambda$	$2.3.4 : p''/\lambda$

Below we give rules and objects associated to the STOP instruction ($1 \leq i \leq n$, $1 \leq j \leq n+1$):

Region	Object(s)
0.	E'_j, E'
1.	E_j, E_Y
2.	$E, S, X_{pqs}, X_{pqs}, X_{p^0}, X_{p^0}, X_q$

Channel	*Rules*	
$(1,0)$	$3.1.1 : EE_Y/\lambda$	$3.1.2 : \lambda/E_YY$
	$3.1.3 : SE_1/\lambda$	$3.1.4 : E'_iE_{i+1}/\lambda$
	$3.1.5 : X_qq/\lambda$	$3.1.6 : X_{pqs}Z_{pqs}/\lambda$
	$3.1.7 : X_{p^0}p^0/\lambda$	
$(1,2)$	$3.2.1 : \lambda/q_fE$	$3.2.2 : \lambda/E'S$
	$3.2.3 : \lambda/E_ia_i$	$3.2.4 : \lambda/E'_iV$
	$3.2.5 : \lambda/E_{n+1}E$	
$(2,0)$	$3.3.1 : \lambda/EE'$	$3.3.2 : \lambda/E_jE'_j$
	$3.3.3 : VE_{n+1}/\lambda$	

We organize system Π as in Lemma 5.5.1. Region 2 contains the current

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

configuration of machine M. All regions contain additional symbols used for the simulation of M.

The simulation of M is similar to Lemma 5.5.1. The incrementing of a register is done directly using the symbol A^+_{pq}. The decrementing operation based on same ideas as for Lemma 5.5.1. More exactly, the signal that the system has to perform the decrementing operation is duplicated (p' and p'') and one symbol tries to decrement the corresponding register, while the other one is delayed in order to make the zero check. The key point consists in the fact that if the register is non-empty, then rules 2.2.2 and 2.2.3 are applied simultaneously and symbols D_{pqs} and Z_{pqs} cannot meet in region 2. Contrarily, if the corresponding register is empty, then these symbols meet in region 2 permitting the simulation of the corresponding branch.

The key difference from the previous lemma consists in the termination procedure. Since the system contains a lot of additional symbols, in particular in region 2, this region must be cleaned at the end of the computation avoiding at the same time possible conflicts with other rules. This is done in several stages. More precisely, the main problem is to move symbols p' from region 2 to region 1. In order to solve this, it is sufficient to disable the group of rules 2.1.1. This might be done in several stages.

1. Move symbol Y to region 1 (thus disabling rule 0.3.1).
2. Move all state symbols (q) to region 0 (disabling rule 2.1.6).
3. Move symbols Z'_{pqs} to region 1 (disabling rule 2.1.5).
4. Move symbols Z_{pqs} to region 0 (disabling rule 2.2.3).
5. Move symbols p'' to region 1 (disabling rule 2.1.2).
6. Move symbols p^0 to region 0 (disabling rule 2.1.1).
7. Finally move symbols p' to region 1.

Rules 3.2.1, 3.1.1 and 3.1.2 permit to realize the first stage. Rules 3.3.2, 3.2.3, 3.2.4 and 3.1.4 permit to move symbols a_i from region 2 to region 1 one after another. Now the main idea is to use existing rules from the decrementing phase and a special enumeration of symbols in a way that will permit to accomplish all stages. First symbols X_q are processed. After being moved to region 1 these symbols move together with corresponding states q to the

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

environment. Since the symbol Y is no more present, symbols q will remain there.

Next, symbols X_{pqs} are processed. These symbols are present in two copies. When the first copy arrives in region 1, it moves the corresponding symbol Z_{pqs} to region 0. After that, symbol Z_{pqs} brings symbol Z'_{pqs} to region 1 (rule 2.1.5). Now, the second copy of X_{pqs} will definitively move symbols Z_{pqs} to region 0, hence realising stages 3 and 4.

In a similar way, symbols X_{p^0} permit to move symbols p'' to region 1 and symbols p^0 to region 0.

Finally, all remaining symbols from region 2 are transferred to region 1 or 0.

Remarks Following same arguments like in Lemma 5.5.1 it is clear that we simulate the behaviour of M.

Now we shall show that the assumption that w_0 is not empty is not necessary. Indeed, it is enough to place initially all symbols s_i that are in w_0, except E', in region 2. In the same region $|w_0| - 1$ symbols U shall be placed. After that, rules $(2, s_iU, 0)$ shall be added. It is clear that during the first step all symbols s_i will move to the environment. The remaining symbol E' shall be placed in region 1, as well as a copy of the symbol U. A similar rule $(1, E'U, 0)$ will move E' to the environment at the first step of the computation. Finally, in order to avoid a possible application of rules 2.2.5 and 3.2.4, the symbol V shall be initially placed in region 1. □

Theorem 5.5.2 $NOtP_2(sym_2) = NRE$.

Proof. It is easy to observe that the system of previous lemma may simulate a non-deterministic register machine. Indeed, in order to simulate a rule $p : (A(A), q, s)$ of such machine, we use the same rules and objects as for rule $p : (A(A), q)$. We only need to add a rule $1.1.1' : \lambda / A^+_{pq} s$ to the channel $(1, 0)$. □

5.6 Two Membranes

Like in the tissue case, it is possible to reduce the number of membranes to two and still obtain computational completeness. However, the price to pay is a few additional objects in the elementary membrane.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

5.6.1 Symport / Antiport of Weight One

We now show that two membranes are enough to obtain computational completeness with symport / antiport rules of minimal size 1 with only three additional objects remaining in halting computations.

Theorem 5.6.1 $N_3OP_2(sym_1, anti_1) = N_3RE$.

Proof. We simulate a counter automaton $M = (d, Q, q_0, q_f, P)$ which starts with empty counters. We also suppose that all instructions from P are labelled in a one-to-one manner with elements of $\{1, \ldots, n\} = I$; I is the disjoint union of $\{n\}$ as well as I_+, I_-, and $I_{=0}$ where by I_+, I_-, and $I_{=0}$ we denote the set of labels for the "increment", "decrement", and "test for zero" instructions, respectively. Additionally we suppose, without loss of generality, that on the first counter of the counter automaton M only "increment" instructions - of the form $(q_i \to q_l, 1+)$ - are operating.

We construct the P system Π_1 as follows:

$$\begin{aligned}
\Pi_1 &= (O, E, [_1\, [_2\ \ ]_2\,]_1, w_1, w_2, E, R_1, R_2, 2),\\
O &= E \cup \{I_c, q_0', F_1, F_2, F_3, F_4, F_5, \#_1, \#_2, b_j, b_j' \mid j \in I\},\\
E &= Q \cup \{a_j, a_j', a_j'' \mid j \in I\} \cup C \cup \{F_2, F_3, F_4, F_5\},
\end{aligned}$$

$$\begin{aligned}
w_1 &= q_0' I_c \#_1 \#_1 \#_2 \#_2,\\
w_2 &= F_1 F_1 F_1 \prod_{j \in I} b_j \prod_{j \in I} b_j',\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,f},\ 1 \le i \le 2.
\end{aligned}$$

The functioning of this system may be split into two stages:

1. simulating the instructions of the counter automaton.
2. terminating the computation.

We code the counter automaton as follows:

Region 1 will hold the current state of the automaton, represented by a symbol $q_i \in Q$; region 2 will hold the value of all counters, represented by the number of occurrences of symbols $c_k \in C$, $k \in D$, where $D = \{1, ..., d\}$. We also use the following idea realized by the phase " START" below: from

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

the environment, we bring symbols c_k into region 1 all the time during the computation. This process may only be stopped if all stages finish correctly; otherwise, the computation will never stop.

We split our proof into several parts that depend on the logical separation of the behavior of the system. We will present the rules and the initial symbols for each part, but we remark that the system we present is the union of all these parts. The rules R_i are given by three phases:

1. START (stage 1);
2. RUN (stage 1);
3. END (stage 2).

The parts of the computations illustrated in the following describe different stages of the evolution of the P system given in the corresponding theorem. For simplicity, we focus on explaining a particular stage and omit the objects that do not participate in the evolution at that time. Each rectangle represents a membrane, each variable represents a copy of an object in a corresponding membrane (symbols outside of the outermost rectangle are found in the environment). In each step, the symbols that will evolve (will be moved) are written in boldface. The labels of the applied rules are written above the symbol $\Rightarrow$.

1. START.

$$\begin{aligned} R_{1,s} &= \{\texttt{1s1} : (I_c, in), \texttt{1s2} : (I_c, out; c_k, in), \texttt{1s3} : (c_k, out) \mid c_k \in C\} \\ &\cup \{\texttt{1s4} : (q_0', out; q_0, in)\}, \\ R_{2,s} &= \emptyset. \end{aligned}$$

Symbol I_c brings one symbol c_k from the environment into region 1 (rules `1s1, 1s2`), where it may be used immediately during the simulation of the " increment" instruction and then moved to region 2. Otherwise symbol c_k returns to the environment (rule `1s3`). Rule `1s4` is used for synchronizing the appearance of the symbols c_k and q_i in region 1.

We illustrate the beginning of the computation as follows:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\mathbf{c_{k_1}}\mathbf{q_0}a_jc_{k_2}\boxed{\mathbf{q_0'I_c}\boxed{b_j}} \Rightarrow^{\mathtt{1s2,1s4}} \mathbf{I_c}q_0'\mathbf{a_j}c_{k_2}\boxed{\mathbf{q_0c_{k_1}}\boxed{b_j}} \Rightarrow^{\mathtt{1s1,1s3,1r1}}$$

$$q_0'q_0c_{k_1}\mathbf{c_{k_2}}\boxed{\mathbf{a_jI_c}\boxed{\mathbf{b_j}}} \Rightarrow^{\mathtt{1s2,2r1}} q_0'q_0c_{k_1}\mathbf{I_c}\boxed{c_{k_2}\mathbf{b_j}\boxed{\mathbf{a_j}}} \cdots$$

2. RUN.

$$\begin{aligned}
R_{1,r} &= \{\mathtt{1r1} : (q_i, out; a_j, in) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, =0\}\} \\
&\cup \{\mathtt{1r2} : (b_j, out; a_j', in), \mathtt{1r3} : (a_j, out; b_j, in), \\
&\quad \mathtt{1r4} : (\#_1, out; b_j, in) \mid j \in I\} \\
&\cup \{\mathtt{1r5} : (a_j', out; a_j'', in) \mid j \in I_+ \cup I_-\} \cup \{\mathtt{1r6} : (\#_1, out; \#_1, in)\} \\
&\cup \{\mathtt{1r7} : (b_j', out; a_j'', in), \mathtt{1r8} : (a_j', out; b_j', in), \\
&\quad \mathtt{1r9} : (\#_1, out; b_j', in) \mid j \in I_{=0}\} \\
&\cup \{\mathtt{1r10} : (a_j'', out, q_l, in) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, =0\}\} \\
&\cup \{\mathtt{1r11} : (b_j, out), \mathtt{1r12} : (b_j', out) \mid j \in I\}, \\
R_{2,r} &= \{\mathtt{2r1} : (b_j, out; a_j, in) \mid j \in I\} \\
&\cup \{\mathtt{2r2} : (a_j, out; c_k, in) \mid (j : q_i \to q_l, k+) \in P\} \\
&\cup \{\mathtt{2r3} : (a_j', in) \mid j \in I_+\} \\
&\cup \{\mathtt{2r4} : (a_j', out; b_j, in) \mid j \in I_+ \cup I_-\} \\
&\cup \{\mathtt{2r5} : (a_j, out) \mid j \in I_- \cup I_{=0}\} \\
&\cup \{\mathtt{2r6} : (c_k, out; a_j', in) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{-, =0\}\} \\
&\cup \{\mathtt{2r7} : (b_j', out; b_j, in), \mathtt{2r8} : (b_j', in) \mid j \in I_{=0}\} \\
&\cup \{\mathtt{2r9} : (a_j, out; \#_2, in) \mid j \in I_+\} \cup \{\mathtt{2r10} : (\#_2, out; \#_2, in)\}.
\end{aligned}$$

" Increment" instruction:

$$\mathbf{a_j}a_j'a_j''q_l\boxed{\mathbf{q_i}c_k\#_1\#_1\boxed{b_j}} \Rightarrow^{\mathtt{1r1}} a_j'a_j''q_iq_l\boxed{\mathbf{a_j}c_k\#_1\#_1\boxed{\mathbf{b_j}}} \Rightarrow^{\mathtt{2r1}}$$

$$\mathbf{a_j'}a_j''q_iq_l\boxed{\mathbf{b_jc_k}\#_1\#_1\boxed{\mathbf{a_j}}} \Rightarrow^{\mathtt{1r2,2r2}} \mathbf{b_j}a_j''q_iq_l\boxed{\mathbf{a_ja_j'}\#_1\#_1\boxed{c_k}}$$

Now there are two possibilities: we may either apply
a) rule 1r5 or
b) rule 2r3.

It is easy to see that **case a)** leads to an infinite computation:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\mathbf{b_j a_j''} q_i q_l \boxed{\mathbf{a_j a_j'} \#_1 \#_1 \boxed{c_k}} \Rightarrow^{\texttt{1r5,1r3}}$$

$$a_j \mathbf{a_j'} q_i \mathbf{q_l} \boxed{\mathbf{b_j a_j''} \#_1 \#_1 \boxed{c_k}} \Rightarrow^{\texttt{1r2,1r10}} a_j \mathbf{b_j} q_i \mathbf{a_j''} \boxed{\mathbf{a_j' q_l} \#_1 \#_1 \boxed{c_k}}$$

After that rule `1r4` will eventually be applied, object $\#_1$ will be moved to the environment and then applying rule `1r6` leads to an infinite computation.

Now let us consider **case b)**:

$$\mathbf{b_j} a_j'' q_i q_l \boxed{\mathbf{a_j a_j'} \#_1 \#_1 \boxed{c_k}} \Rightarrow^{\texttt{1r3,2r3}} a_j a_j'' q_i q_l \boxed{\mathbf{b_j} \#_1 \#_1 \boxed{\mathbf{a_j'} c_k}}$$

We cannot apply rule `1r2` as this leads to an infinite computation (see above). Hence, rule `2r4` has to be applied:

$$a_j a_j'' q_i q_l \boxed{\mathbf{b_j} \#_1 \#_1 \boxed{\mathbf{a_j'} c_k}} \Rightarrow^{\texttt{2r4}} a_j \mathbf{a_j''} q_i q_l \boxed{\mathbf{a_j'} \#_1 \#_1 \boxed{b_j c_k}} \Rightarrow^{\texttt{1r5}}$$

$$a_j a_j' q_i \mathbf{q_l} \boxed{\mathbf{a_j''} \#_1 \#_1 \boxed{b_j c_k}} \Rightarrow^{\texttt{1r10}} a_j a_j' a_j'' q_i \boxed{\mathbf{q_l} \#_1 \#_1 \boxed{b_j c_k}}$$

In that way, q_i is replaced by q_l and c_k is moved from region 1 into region 2.

" Decrement" instruction:

$$\mathbf{a_j} a_j' a_j'' q_l \boxed{\mathbf{q_i} \#_1 \#_1 \boxed{b_j c_k}} \Rightarrow^{\texttt{1r1}} a_j' a_j'' q_i q_l \boxed{\mathbf{a_j} \#_1 \#_1 \boxed{\mathbf{b_j} c_k}} \Rightarrow^{\texttt{2r1}}$$

$$\mathbf{a_j'} a_j'' q_i q_l \boxed{\mathbf{b_j} \#_1 \#_1 \boxed{\mathbf{a_j} c_k}} \Rightarrow^{\texttt{1r2,2r5}} \mathbf{b_j} a_j'' q_i q_l \boxed{\mathbf{a_j a_j'} \#_1 \#_1 \boxed{c_k}} \Rightarrow^{\texttt{1r3,2r6}}$$

$$a_j a_j'' q_i q_l \boxed{\mathbf{b_j} c_k \#_1 \#_1 \boxed{\mathbf{a_j'}}} \Rightarrow^{\texttt{2r4}} a_j \mathbf{a_j''} q_i q_l \boxed{\mathbf{a_j'} c_k \#_1 \#_1 \boxed{b_j}} \Rightarrow^{\texttt{1r5}}$$

$$a_j a_j' q_i \mathbf{q_l} \boxed{\mathbf{a_j''} c_k \#_1 \#_1 \boxed{b_j}} \Rightarrow^{\texttt{1r10}} a_j a_j' a_j'' q_i \boxed{\mathbf{q_l} c_k \#_1 \#_1 \boxed{b_j}}$$

In the way described above, q_i is replaced by q_l and c_k is removed from region 2 to region 1.

" Test for zero" instruction:

q_i is replaced by q_l if there is no c_k in region 2, otherwise a_j' in region 1 exchanges with c_k in region 2 and the computation will never stop.

(i) *There is no c_k in region 2:*

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\mathbf{a_j}a'_ja''_jq_l\boxed{\mathbf{q_i}\#_1\#_1\boxed{b_jb'_j}} \Rightarrow^{\texttt{1r1}} a'_ja''_jq_iq_l\boxed{\mathbf{a_j}\#_1\#_1\boxed{\mathbf{b_j}b'_j}} \Rightarrow^{\texttt{2r1}}$$

$$a'_ja''_jq_iq_l\boxed{\mathbf{b_j}\#_1\#_1\boxed{\mathbf{a_j}b'_j}}$$

Now there are two possibilities: we apply either

a) rule `2r7` or

b) rule `1r2`.

It is easy to see that **case a)** leads to an infinite computation:

$$a'_ja''_jq_iq_l\boxed{\mathbf{b_j}\#_1\#_1\boxed{\mathbf{a_jb'_j}}} \Rightarrow^{\texttt{2r7,2r5}} a'_ja''_jq_iq_l\boxed{\mathbf{a_jb'_j}\#_1\#_1\boxed{\mathbf{b_j}}} \Rightarrow^{\texttt{2r1,2r8}}$$

$$a'_ja''_jq_iq_l\boxed{\mathbf{b_j}\#_1\#_1\boxed{\mathbf{a_jb'_j}}} \Rightarrow^{\texttt{2r7,2r5}} \cdots \Rightarrow^{\texttt{2r1,2r8}}$$

$$\mathbf{a'_j}a''_jq_iq_l\boxed{\mathbf{b_j}\#_1\#_1\boxed{\mathbf{a_j}b'_j}} \Rightarrow^{\texttt{1r2,2r5}}$$

$$\mathbf{b_j}a''_jq_iq_l\boxed{\mathbf{a_j}a'_j\#_1\#_1\boxed{b'_j}} \Rightarrow^{\texttt{1r3}} a_ja''_jq_iq_l\boxed{\mathbf{b_j}a'_j\#_1\#_1\boxed{b'_j}}$$

Again there are two possibilities: we can apply either

c) rule `1r2` or

d) rule `2r7`.

Case c) leads to an infinite computation (rules `1r4` and `1r6`).

Now let us consider **case d)**:

$$a_ja''_jq_iq_l\boxed{\mathbf{b_j}a'_j\#_1\#_1\boxed{\mathbf{b'_j}}} \Rightarrow^{\texttt{2r7}} a_j\mathbf{a''_j}q_iq_l\boxed{\mathbf{b'_j}a'_j\#_1\#_1\boxed{b_j}} \Rightarrow^{\texttt{1r7}}$$

$$a_j\mathbf{b'_j}q_i\mathbf{q_l}\boxed{\mathbf{a''_ja'_j}\#_1\#_1\boxed{b_j}} \Rightarrow^{\texttt{1r8,1r10}} a_ja'_ja''_iq_i\boxed{\mathbf{q_l}\mathbf{b'_j}\#_1\#_1\boxed{b_j}}$$

There are two possibilities: we can apply either

e) rule `1r7` or

f) rule `2r8`.

Case e) leads to infinite computation (rules `1r9` and `1r6`).

In **case f)**, the object b'_j comes back to region 2.

(ii) *There is some c_k in region 2:*

Consider again **case d)**:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$a_j a''_j q_i q_l \boxed{\mathbf{b_j} \mathbf{a'_j} \#_1 \#_1 \boxed{\mathbf{b'_j} \mathbf{c_k}}} \Rightarrow^{\texttt{2r7,2r6}} a_j \mathbf{a''_j} q_i q_l \boxed{\mathbf{b'_j} c_k \#_1 \#_1 \boxed{a'_j b_j}} \Rightarrow^{\texttt{1r7}}$$

$$a_j \mathbf{b'_j} q_i \mathbf{q_l} \boxed{\mathbf{a''_j} c_k \#_1 \#_1 \boxed{a'_j b_j}} \Rightarrow^{\texttt{1r9,1r10}} a_j a''_j \#_1 q_i \boxed{\mathbf{q_l} \mathbf{b'_j} c_k \#_1 \boxed{a'_j b_j}}$$

Now the application of rule `1r6` leads to an infinite computation.

Finally, let us notice that applying the rules `1r11` and `1r12` during the phase RUN leads to infinite computation. Hence, we model correctly the "test for zero" instruction.

3. END.

$$\begin{aligned} R_{1,f} &= \{\texttt{1f1} : (F_1, out; F_2, in), \texttt{1f2} : (F_2, out; F_3, in), \\ &\quad \texttt{1f3} : (F_3, out; F_4, in), \texttt{1f4} : (F_4, out; F_5, in), \\ R_{2,f} &= \{\texttt{2f1} : (F_1, out; q_f, in), \texttt{2f2} : (q_f, out; I_c, in), \\ &\quad \texttt{2f3} : (q_f, out; \#_1, in), \texttt{2f4} : (q_f, out; \#_2, in), \texttt{2f5} : (F_5, out), \\ &\quad \texttt{2f6} : (b_j, out; F_5, in), \texttt{2f7} : (b'_j, out; F_5, in)\}. \end{aligned}$$

We illustrate the end of computations as follows:

$$F_2 F_3 F_4 F_5 I_c c_{k_1} c_{k_2} \boxed{q_f \#_1 \#_1 \#_2 \#_2 \boxed{F_1 F_1 F_1 b_{j_1} b'_{j_2}}}$$

$$\Rightarrow^{\texttt{2f1,1s1}}$$

$$F_2 F_3 F_4 F_5 c_{k_1} c_{k_2} \boxed{I_c \#_1 \#_1 \#_2 \#_2 F_1 \boxed{q_f F_1 F_1 b_{j_1} b'_{j_2}}}$$

$$\Rightarrow^{\texttt{2f3,1s2,1f1}}$$

$$F_2 F_3 F_4 F_5 I_c c_{k_2} F_1 \boxed{F_2 c_{k_1} \#_1 \#_2 \#_2 q_f \boxed{\#_1 F_1 F_1 b_{j_1} b'_{j_2}}}$$

$$\Rightarrow^{\texttt{1s1,1s4,1f2,2f1}}$$

$$F_2 F_3 F_4 F_5 c_{k_1} c_{k_2} F_1 \boxed{F_3 I_c \#_1 \#_2 \#_2 F_1 \boxed{q_f \#_1 F_1 b_{j_1} b'_{j_2}}}$$

$$\Rightarrow^{\texttt{1s2,1f1,1f3,2f3}}$$

$$F_2 F_3 F_4 F_5 c_{k_1} I_c F_1 F_1 \boxed{F_2 F_4 c_{k_2} \#_2 \#_2 q_f \boxed{\#_1 \#_1 F_1 b_{j_1} b'_{j_2}}}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\Rightarrow^{\texttt{1s1,1s4,1f2,1f4,2f1}}$$

$$F_2F_3F_4F_5c_{k_1}c_{k_2}F_1F_1\boxed{F_3F_5I_c\#_2\#_2F_1\boxed{q_f\#_1\#_1b_{j_1}b'_{j_2}}}$$

Notice that now rule `2f2` will be applied eventually, as otherwise the application of rule `2f4` will lead to an infinite computation (rule `2r10`). Hence, we continue as follows:

$$F_2F_3F_4F_5c_{k_1}c_{k_2}F_1F_1\boxed{F_3F_5I_c\#_2\#_2F_1\boxed{q_f\#_1\#_1b_{j_1}b'_{j_2}}}$$

$$\Rightarrow^{\texttt{1f1,1f3,2f2,2f6}}$$

$$F_2F_3F_4F_5c_{k_1}c_{k_2}F_1F_1F_1\boxed{F_2F_4\#_2\#_2b_{j_1}q_f\boxed{I_c\#_1\#_1F_5b'_{j_2}}}$$

$$\Rightarrow^{\texttt{1f2,1f4,1r11,2f5}}$$

$$F_2F_3F_4F_5c_{k_1}c_{k_2}F_1F_1F_1b_{j_1}\boxed{F_3F_5F_5\#_2\#_2q_f\boxed{I_c\#_1\#_1b'_{j_2}}}$$

We continue in this manner until all objects $b_j, b'_j,\ j \in I$ from the elementary membrane 2 have been moved to the environment. Notice that the result in the elementary membrane 2 (multiset c_1^t) cannot be changed during phase END, as object I_c now is situated in the elementary membrane and cannot bring symbols c_1 from the environment. Recall that the counter automaton can only increment the first counter c_1, so all other computations of P system Π_1 cannot change the number of symbols c_1 in the elementary membrane. Thus, at the end of a terminating computation, in the elementary membrane there are the result (multiset c_1^t) and only the three additional objects $I_c, \#_1, \#_1$. □

5.6.2 Symport of Weight Two

A "dual" class of systems with minimal cooperation is the class where two objects are moved across the membrane in the same direction rather than in the opposite ones. We now prove a similar result for this class using six additional symbols.

Theorem 5.6.2 $N_6OP_2(sym_2) = N_6RE$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Proof. As in the proof of Theorem 5.6.1 we simulate a counter automaton $M = (d, Q, q_0, q_f, P)$ that starts with empty counters. Again we suppose that all instructions from P are labelled in a one-to-one manner with elements of $\{1, \ldots, n\} = I$ and that I is the disjoint union of $\{n\}$ as well as I_+, I_-, and $I_{=0}$ where by I_+, I_-, and $I_{=0}$ we denote the set of labels for the "increment", "decrement", and "test for zero" instructions, respectively. Moreover, we define $I' = \{1, 2, \ldots, n+4\}$, $Q_k = \{q_{i,k}\}$, $1 \leq k \leq 5$, $i \in K$, $K = \{0, 1, \ldots, f\}$, and $C = \{c_i \mid 1 \leq i \leq d\}$.

We construct the P system Π_2 as follows:

$$
\begin{aligned}
\Pi_2 &= (O, [_1\ [_2\ \ ]_2\ ]_1, w_1, w_2, E, R_1, R_2, 2),\\
O &= \{\#_0, \#_1, \#_2, \$_1, \$_2, \$_3, \hat{a}, \hat{b}, I_c\} \cup \{a_k \mid 1 \leq k \leq 5\} \cup Q \cup \bigcup_{1 \leq k \leq 5} Q_k\\
&\cup\ C \cup \{a_j, a'_j, \check{a}_j, \hat{a}_j, b_j, d_j, d'_j, d''_j \mid j \in I\} \cup \{e_t, h_t \mid t \in I'\}\\
E &= \{a_1, a_3, a_5, \#_0\} \cup \{a_j, a'_j \mid j \in I\} \cup \{h_t \mid t \in I'\} \cup Q \cup Q_2 \cup Q_4 \cup C,\\
w_1 &= \#_1 \hat{a} \hat{b} a_2 a_4 \$_3 \prod_{j \in I} (\check{a}_j d'_j d''_j) \prod_{t \in I'} e_t \prod_{i \in K} (\hat{q}_i q_{i,1} q_{i,3} q_{i,5})\\
w_2 &= \#_2 \$_1^{n+1} \$_2 \prod_{j \in I} (\hat{a}_j b_j d_j),\\
R_i &= R_{i,s} \cup R_{i,r} \cup R_{i,f}, i \in \{1, 2\}.
\end{aligned}
$$

The functioning of this system again may be split into two stages:

1. simulating the instructions of the counter automaton;
2. terminating the computation.

We code the counter automaton as in Theorem 5.6.1 above: region 1 will hold the current state of the automaton, represented by a symbol $q_i \in Q$; region 2 will hold the value of all counters, represented by the number of occurrences of symbols $c_k \in C$, $k \in D$, where $D = \{1, ..., d\}$. We also use the following idea (called "*Circle*") realized by phase "START" below: from the environment, we bring symbols c_k into region 1 all the time during the computation. This process may only be stopped if all stages finish correctly; otherwise, the computation will never stop.

We split our proof into several parts that depend on the logical separation of the behavior of the system. We will present the rules and the initial

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

symbols for each part, but we remark that the system that we present is the union of all these parts.

The rules R_i again are given by three phases:

START (stage 1); RUN (stage 1); END (stage 2).

1. START.

$$\begin{aligned} R_{1,s} &= \{\texttt{1s1} : (I_c, out), \texttt{1s2} : (I_c c_k, in), \texttt{1s3} : (c_k, out) \mid k \in D\}, \\ R_{2,s} &= \emptyset. \end{aligned}$$

Symbol I_c brings one symbol $c \in C$ from the environment into region 1 (rules `1s1, 1s2`) where it may be used immediately during the simulation of an "increment" instruction and moved to region 2. Otherwise symbol c returns to the environment (rule `1s3`).

2. RUN.

$$\begin{aligned} R_{1,r} &= \{\texttt{1r1} : (q_i \hat{q}_i, out) \mid i \in K\} \\ &\cup \{\texttt{1r2} : (a_j \hat{q}_i, in) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, =0\}, k \in D\} \\ &\cup \{\texttt{1r3} : (a_j \hat{a}, out) \mid j \in I_+ \cup I_-\} \cup \{\texttt{1r4} : (a_j \hat{b}, out) \mid j \in I_{=0}\} \\ &\cup \{\texttt{1r5} : (\#_2, out), \texttt{1r6} : (\#_2, in)\} \cup \{\texttt{1r7} : (b_j \check{a}_j, out) \mid j \in I\} \\ &\cup \{\texttt{1r8} : (b_j \#_1, out) \mid j \in I\} \cup \{\texttt{1r9} : (\hat{a}_j \#_1, out) \mid j \in I\} \\ &\cup \{\texttt{1r10} : (\#_0 \#_1, in), \texttt{1r11} : (\#_0 \hat{b}, in)\} \cup \{\texttt{1r12} : (a'_j b_j, in) \mid j \in I\} \\ &\cup \{\texttt{1r13} : (\hat{a} a_1, in), \texttt{1r14} : (a_1 a_2, out), \texttt{1r15} : (a_2 a_3, in)\} \\ &\cup \{\texttt{1r16} : (a_3 a_4, out), \texttt{1r17} : (a_4 a_5, in), \texttt{1r18} : (a_5, out)\} \\ &\cup \{\texttt{1r19} : (a'_j q_{l,1}, out) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{+, -, =0\}, k \in D\} \\ &\cup \{\texttt{1r20} : (q_{i,1} q_{i,2}, in), \texttt{1r21} : (q_{i,2} q_{i,3}, out), \texttt{1r22} : (q_{i,3} q_{i,4}, in) \mid i \in K\} \\ &\cup \{\texttt{1r23} : (q_{i,4} q_{i,5}, out), \texttt{1r24} : (q_{i,5} q_i, in) \mid i \in K\} \\ &\cup \{\texttt{1r25} : (d_j \hat{a}, out), \texttt{1r26} : (d_j \#_0, in) \mid j \in I_+ \cup I_-\} \\ &\cup \{\texttt{1r27} : (d_j \check{a}_j, in) \mid j \in I\} \cup \{\texttt{1r28} : (d_j \#_1, out) \mid j \in I_+ \cup I_-\} \\ &\cup \{\texttt{1r29} : (d_j d'_j, out) \mid j \in I_{=0}\} \cup \{\texttt{1r30} : (d'_j \hat{b}, in) \mid j \in I_{=0}\}, \end{aligned}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned}
R_{2,r} &= \{\texttt{2r1} : (a_j\check{a}_j, in) \mid j \in I\} \cup \{\texttt{2r2} : (b_j\check{a}_j, out) \mid j \in I\} \\
&\cup \{\texttt{2r3} : (a_jc_k, out) \mid (j : q_i \to q_l, k\gamma) \in P, \gamma \in \{-, = 0\}, k \in D\} \\
&\cup \{\texttt{2r4} : (a_j\#_2, out) \mid j \in I_-\} \cup \{\texttt{2r5} : (a_j\hat{a}_j, out) \mid j \in I_+\} \\
&\cup \{\texttt{2r6} : (\#_0, in), \texttt{2r7} : (\#_0, out)\} \\
&\cup \{\texttt{2r8} : (c_k\hat{a}_j, in) \mid (j : q_i \to q_l, k+) \in P, k \in D\} \\
&\cup \{\texttt{2r9} : (a'_jb_j, in) \mid j \in I\} \cup \{\texttt{2r10} : (a'_jd_j, out) \mid j \in I\} \\
&\cup \{\texttt{2r11} : (d_ja_5, in) \mid j \in I_+ \cup I_-\} \cup \{\texttt{2r12} : (a_5, out)\} \\
&\cup \{\texttt{2r13} : (d_jd''_j, in) \mid j \in I_{=0}\} \cup \{\texttt{2r14} : (a_jd''_j, out) \mid j \in I_{=0}\}.
\end{aligned}$$

"Increment" instruction:

$$a_jc\boxed{\mathbf{I_cq_i\hat{q}_i}\check{a}_j\hat{a}\boxed{b_j\hat{a}_j}} \Rightarrow^{\texttt{1r1,1s1}} q_i\mathbf{\hat{q}_ia_jI_cc}\boxed{\check{a}_j\hat{a}\boxed{b_j\hat{a}_j}} \Rightarrow^{\texttt{1r2,1s2}}$$

$$q_i\boxed{I_cc\hat{q}_ia_j\check{a}_j\hat{a}\boxed{b_j\hat{a}_j}}, \text{ where } c \in C$$

Now there are two variants of computations (depending on the application of rule `2r1` or rule `1r3`). It is easy to see that the application of rule `1r3` leads to an infinite computation (by *"Circle"*). Consider applying rule `2r1`:

$$q_ic_k\boxed{\mathbf{I_cc}\hat{q}_i\mathbf{a_j\check{a}_j}\hat{a}\boxed{b_j\hat{a}_j}} \Rightarrow^{\texttt{2r1,1s1,1s3}}$$

$$q_i\mathbf{I_cc_k}c\boxed{\hat{q}_i\hat{a}\boxed{\mathbf{b_j\check{a}_ja_j\hat{a}_j}}} \Rightarrow^{\texttt{2r2,2r5,1s2}}$$

$$q_ic\boxed{I_cc_k\hat{q}_i\hat{a}b_j\check{a}_ja_j\hat{a}_j\boxed{}}$$

Notice that object $\hat{a}_j$ cannot be idle, as the application of the rules `1r9`, `1r10`, `2r6`, `2r7` leads to an infinite computation. Hence, rule `2r8` will be applied and object c_k will be moved to region 2 (thus, we increase the number of objects c_k in region 2 by one and model the increment instruction of the counter automaton). In an analogous way, object b_j cannot be idle, as applying rules `1r8`, `1r10`, `2r6`, `2r7` leads to an infinite computation. Thus, rule `2r1` cannot be applied and rule `1r7` will eventually be applied.

$$ca'_ja_1a_3a_5\boxed{\mathbf{I_cc_k}\hat{q}_i\hat{a}\mathbf{b_j\check{a}_ja_j\hat{a}_j}a_2a_4q_{l,1}\boxed{}}$$

$$\mathbf{I_cca'_jb_j}\check{a}_ja_j\mathbf{\hat{a}a_1}a_3a_5\boxed{\hat{q}_ia_2a_4q_{l,1}\boxed{\hat{a}_jc_k}} \Rightarrow^{\texttt{1r12,1r13,1s2}}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$\check{a}_j a_j a_3 a_5 \boxed{I_c c \hat{q}_i \hat{a} a_1 a_2 a_4 q_{l,1} a'_j b_j \boxed{\hat{a}_j c_k}}$

Notice that applying rule `1r19` leads to an infinite computation, as object b_j cannot be idle. Thus, rule `2r9` will eventually be applied.

$\check{a}_j a_j a_3 a_5 q_{l,2} q_{l,4} \boxed{\mathbf{I_c c} \hat{q}_i \hat{a} \mathbf{a_1 a_2} a_4 q_{l,1} \mathbf{a'_j b_j} q_{l,3} q_{l,5} \boxed{d_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{2r9,1r14,1s1,1s3}}$

$\mathbf{I_c c} \check{a}_j a_j a_1 \mathbf{a_2 a_3} a_5 q_{l,2} q_{l,4} \boxed{\hat{q}_i \hat{a} a_4 q_{l,1} q_{l,3} q_{l,5} \boxed{\mathbf{d_j a'_j} b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{2r10,1r15,1s2}}$

$\check{a}_j a_j a_1 a_5 q_{l,2} q_{l,4} \boxed{\mathbf{I_c c} \hat{q}_i a_2 \mathbf{a_3 a_4 \hat{a} d_j a'_j q_{l,1}} q_{l,3} q_{l,5} \boxed{b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{1r19,1r25,1r16,1s1,1s3}}$

$\mathbf{I_c c} a_j \mathbf{\check{a}_j d_j \hat{a} a_1} a_3 \mathbf{a_4 a_5} a'_j \mathbf{q_{l,1} q_{l,2}} q_{l,4} \boxed{\hat{q}_i a_2 q_{l,3} q_{l,5} \boxed{b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{1r27,1r13,1r17,1r20,1s2}}$

$a_j a_3 a'_j q_{l,4} \boxed{I_c c \hat{q}_i \hat{a} a_1 a_2 a_4 \check{a}_j d_j a_5 q_{l,1} q_{l,2} q_{l,3} q_{l,5} \boxed{b_j \hat{a}_j c_k}}$

Now we can apply the rules `1r25`, `1r18` or `2r11`. It is easy to see that applying rule `1r25` leads to an infinite computation (rules `1r26`, `2r6`, `2r7`), which is true for rule `1r18`, too (rules `1r28`, `1r10`, `2r6`, `2r7`). Hence, now consider applying rule `2r11`.

$a_j a_3 a'_j q_{l,4} q_l \boxed{\mathbf{I_c c} \hat{q}_l \hat{q}_i \hat{a} \mathbf{a_1 a_2} a_4 \check{a}_j \mathbf{d_j a_5} q_{l,1} \mathbf{q_{l,2} q_{l,3}} q_{l,5} \boxed{b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{2r11,1r21,1r14,1s1,1s3}}$

$\mathbf{I_c c} a_j a_1 \mathbf{a_2 a_3} a'_j q_{l,2} \mathbf{q_{l,3} q_{l,4}} q_l \boxed{\hat{q}_l \hat{q}_i \hat{a} a_4 \check{a}_j q_{l,1} q_{l,5} \boxed{d_j \mathbf{a_5} b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{2r12,1r15,1r22,1s2}}$

$a_j a_1 a'_j q_{l,2} q_l \boxed{\mathbf{I_c c} \hat{q}_l \hat{q}_i \hat{a} a_2 \mathbf{a_3 a_4 a_5} \check{a}_j q_{l,1} q_{l,3} \mathbf{q_{l,4} q_{l,5}} \boxed{d_j b_j \hat{a}_j c_k}}$

$\Rightarrow^{\texttt{1r16,1r18,1r23,1s1,1s3}}$

$\mathbf{I_c c} a_j a_1 a_3 \mathbf{a_4 a_5} a'_j q_{l,2} q_{l,4} \mathbf{q_{l,5} q_l} \boxed{\hat{q}_l \hat{q}_i \hat{a} a_2 \check{a}_j q_{l,1} q_{l,3} \boxed{d_j b_j \hat{a}_j c_k}}$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\Rightarrow^{\texttt{1r17,1r24,1s2}}$$

$$a_j a_1 a_3 a'_j q_{l,2} q_{l,4} \boxed{\mathbf{I_c c q_l \hat{q}_l} \hat{q}_i \hat{a} a_2 a_4 \mathbf{a_5} \check{a}_j q_{l,1} q_{l,3} q_{l,5} \boxed{d_j b_j \hat{a}_j c_k}}$$

$$\Rightarrow^{\texttt{1r1,1r18,1s1,1s3}}$$

$$I_c c a_j a_1 a_3 a_5 a'_j q_{l,2} q_{l,4} q_l \hat{q}_l \boxed{\hat{q}_i \hat{a} a_2 a_4 \check{a}_j q_{l,1} q_{l,3} q_{l,5} \boxed{d_j b_j \hat{a}_j c_k}}$$

Thus, we begin a new circle of modelling.

"Decrement" instruction.

If there is an object c_k in region 2, we obtain the following computation:

$$a_j \boxed{\mathbf{q_i \hat{q}_i} \check{a}_j \hat{a} \boxed{b_j c_k \#_2}} \Rightarrow^{\texttt{1r1}} q_i \mathbf{\hat{q}_i a_j} \boxed{\check{a}_j \hat{a} \boxed{b_j c_k \#_2}} \Rightarrow^{\texttt{1r2}}$$

$$q_i \boxed{\hat{q}_i a_j \check{a}_j \hat{a} \boxed{b_j c_k \#_2}}$$

Now there are two variants of computations (depending on the application of rule `2r1` or rule `1r3`). It is easy to see that the application of rule `1r3` leads to an infinite computation (by *"Circle"*). Now consider applying rule `2r1`:

$$q_i \boxed{\hat{q}_i \mathbf{a_j \check{a}_j} \hat{a} \boxed{b_j c_k \#_2}} \Rightarrow^{\texttt{2r1}} q_i \boxed{\hat{q}_i \hat{a} \boxed{\mathbf{b_j \check{a}_j a_j c_k} \#_2}} \Rightarrow^{\texttt{2r2,2r3}}$$

$$q_i \boxed{\hat{q}_i b_j \check{a}_j \hat{a} a_j c_k \boxed{\#_2}}$$

Thus, object c_k is moved from region 2 to region 1 (thus, we decrease the number of objects c_k in region 2 by one and model the "decrement" instruction of the counter automaton).

The case when there is no object c_k in region 2 leads to an infinite computation (rules `2r4`, `1r5`, `1r6`), hence, again we correctly model the "decrement" instruction. The further behavior of the system is the same as in the case of modelling the "increment" instruction.

"Test for zero" instruction:

q_i is replaced by q_l if there is no c_k in region 2 (case a)), otherwise the computation will never stop (case b)).

Case a):

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$a_j\boxed{\mathbf{q_i}\hat{\mathbf{q}}_\mathbf{i}\check{a}_j\hat{b}d'_jd''_j\boxed{b_jd_j\#_2}} \Rightarrow^{\texttt{1r1}} q_i\hat{\mathbf{q}}_\mathbf{i}\mathbf{a_j}\boxed{\check{a}_j\hat{b}d'_jd''_j\boxed{b_jd_j\#_2}} \Rightarrow^{\texttt{1r2}}$$
$$q_i\boxed{\hat{q}_ia_j\check{a}_j\hat{b}d'_jd''_j\boxed{b_jd_j\#_2}}$$

Now there are two variants of computations (depending on the application of rule `2r1` or rule `1r4`). It is easy to see that the application of rule `1r4` leads to an infinite computation (by *"Circle"*). Consider the application of rule `2r1`:

$$q_iq_{l,2}q_{l,4}q_la'_j\boxed{\hat{q}_i\mathbf{a_j}\check{\mathbf{a}}_\mathbf{j}q_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{b_jd_j\#_2}} \Rightarrow^{\texttt{2r1}}$$
$$q_iq_{l,2}q_{l,4}q_la'_j\boxed{\hat{q}_iq_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_j\check{\mathbf{a}}_\mathbf{j}\mathbf{b_j}d_j\#_2}} \Rightarrow^{\texttt{2r2}}$$
$$q_iq_{l,2}q_{l,4}q_la'_j\boxed{\hat{q}_i\check{\mathbf{a}}_\mathbf{j}\mathbf{b_j}q_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_jd_j\#_2}} \Rightarrow^{\texttt{1r7}}$$
$$q_iq_{l,2}q_{l,4}q_l\check{a}_j\mathbf{b_ja'_j}\boxed{\hat{q}_iq_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_jd_j\#_2}} \Rightarrow^{\texttt{1r12}}$$
$$q_iq_{l,2}q_{l,4}q_l\check{a}_j\boxed{\hat{q}_ib_ja'_jq_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_jd_j\#_2}}$$

Again there are two variants of computations, depending on the application of rule `1r19` or rule `2r9`. Notice that applying rule `1r19` leads to an infinite computation, as object b_j cannot be idle (rules `1r8`, `1r10`, `2r6`, `2r7`). Hence, we only consider the case of applying rule `2r9`:

$$q_iq_{l,2}q_{l,4}q_l\check{a}_j\boxed{\hat{q}_i\mathbf{b_ja'_j}q_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_jd_j\#_2}} \Rightarrow^{\texttt{2r9}}$$
$$q_iq_{l,2}q_{l,4}q_l\check{a}_j\boxed{\hat{q}_iq_{l,1}q_{l,3}q_{l,5}\hat{b}d'_jd''_j\boxed{a_jb_j\mathbf{a'_jd_j}\#_2}} \Rightarrow^{\texttt{2r10}}$$
$$q_iq_{l,2}q_{l,4}q_l\check{a}_j\boxed{\hat{q}_ia'_jq_{l,1}q_{l,3}q_{l,5}\hat{b}d_jd'_jd''_j\boxed{a_jb_j\#_2}}$$

Now there are two variants of computations, depending on the application of rule `2r13` and `1r29`. It is easy to see that applying rule `2r14` leads to an infinite computation (rules `2r14`, `1r4`, `1r11`, `2r6`, `2r7`). Hence, consider applying rule `1r29`:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$q_i q_{l,2} q_{l,4} q_l \check{a}_j \boxed{\hat{q}_i \mathbf{a'_j q_{l,1}} q_{l,3} q_{l,5} \hat{b} \mathbf{d_j d'_j} d''_j \boxed{a_j b_j \#_2}} \Rightarrow^{\texttt{1r29,1r19}}$$

$$q_i a'_j \mathbf{q_{l,1} q_{l,2}} q_{l,4} q_l \mathbf{\check{a}_j d_j} d'_j \boxed{\hat{q}_i q_{l,3} q_{l,5} \hat{b} d''_j \boxed{a_j b_j \#_2}} \Rightarrow^{\texttt{1r20,1r27}}$$

$$q_i a'_j q_{l,4} q_l d'_j \boxed{\hat{q}_i q_{l,1} \mathbf{q_{l,2} q_{l,3}} q_{l,5} \hat{b} \check{a}_j \mathbf{d_j d''_j} \boxed{a_j b_j \#_2}} \Rightarrow^{\texttt{1r21,2r13}}$$

$$q_i a'_j q_{l,2} \mathbf{q_{l,3} q_{l,4}} q_l d'_j \boxed{\hat{q}_i q_{l,1} q_{l,5} \hat{b} \check{a}_j \boxed{d_j \mathbf{d''_j a_j} b_j \#_2}} \Rightarrow^{\texttt{1r22,2r14}}$$

$$q_i a'_j q_{l,2} q_l d'_j \boxed{\hat{q}_i q_{l,1} q_{l,3} \mathbf{q_{l,4} q_{l,5}} d''_j \mathbf{a_j \hat{b}} \check{a}_j \boxed{d_j b_j \#_2}} \Rightarrow^{\texttt{1r4,1r23}}$$

$$q_i a'_j q_{l,2} q_{l,4} \mathbf{q_{l,5} q_l} a_j \mathbf{\hat{b} d'_j} \boxed{\hat{q}_i q_{l,1} q_{l,3} d''_j \check{a}_j \boxed{d_j b_j \#_2}} \Rightarrow^{\texttt{1r24,1r30}}$$

$$q_i a'_j q_{l,2} q_{l,4} a_j \boxed{\hat{q}_i q_{l,1} q_{l,3} q_{l,5} q_l \hat{b} d'_j d''_j \check{a}_j \boxed{d_j b_j \#_2}}$$

Thus, q_i is replaced by q_l in region 1.

Case b):

$$a_j \boxed{\mathbf{q_i \hat{q}_i} \check{a}_j \hat{b} \boxed{c_k b_j d_j \#_2}} \Rightarrow^{\texttt{1r1}} q_i \mathbf{\hat{q}_i a_j} \boxed{\check{a}_j \hat{b} \boxed{c_k b_j d_j \#_2}} \Rightarrow^{\texttt{1r2}}$$

$$q_i \boxed{\hat{q}_i a_j \check{a}_j \hat{b} \boxed{c_k b_j d_j \#_2}}$$

Again there are two variants of computations (depending on the application of rule `2r1` or rule `1r4`). It is easy to see that the application of rule `1r4` leads to infinite computation (by *"Circle"*). Consider applying rule `2r1`:

$$q_i \boxed{\hat{q}_i \mathbf{a_j \check{a}_j} \hat{b} \boxed{c_k b_j d_j \#_2}} \Rightarrow^{\texttt{2r1}} q_i \boxed{\hat{q}_i \hat{b} \boxed{\mathbf{c_k a_j \check{a}_j b_j} d_j \#_2}} \Rightarrow^{\texttt{2r2,2r3}}$$

$$q_i \boxed{\hat{q}_i \check{a}_j b_j c_k a_j \hat{b} \boxed{d_j \#_2}}$$

There are two variants of computations, depending on the application of rule `2r1` or rule `1r4`. Notice that they both lead to infinite computations. Indeed, if rule `2r1` will be applied, then rules `1r8`, `1r10`, `2r6`, `2r7` will be applied (applying rules `2r6`, `2r7` leads to an infinite computation). If rule `1r4` will

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

be applied, it again leads to an infinite computation (rules $\texttt{1r11}, \texttt{2r6}, \texttt{2r7}$). Thus, we correctly model a "test for zero" instruction.

3. END.

$$\begin{aligned}
R_{1,f} &= \{\texttt{1f1} : (\$_1 \breve{a}_j, out) \mid j \in I\} \\
&\cup \{\texttt{1f2} : (\$_2 e_1, out), \texttt{1f3} : (\$_1 \$_3, out)\} \\
&\cup \{\texttt{1f4} : (e_t h_t, in) \mid t \in I'\} \\
&\cup \{\texttt{1f5} : (h_t e_{t+1}, out) \mid 1 \leq t \leq n+3\} \\
R_{2,f} &= \{\texttt{2f1} : (q_f, in), \texttt{2f2} : (q_f \$_1, out), \texttt{2f3} : (q_f \$_2, out)\} \\
&\cup \{\texttt{2f4} : (\$_1 \hat{a}, in), \texttt{2f5} : (\$_1 \#_1, in), \texttt{2f6} : (\$_1 I_c, in)\} \\
&\cup \{\texttt{2f7} : (h_{n+4}, in)\} \\
&\cup \{\texttt{2f8} : (h_{n+4} \hat{a}_j, out) \mid j \in I\} \\
&\cup \{\texttt{2f9} : (h_{n+4} b_j, out) \mid j \in I\} \\
&\cup \{\texttt{2f10} : (h_{n+4} d_j, out) \mid j \in I\}
\end{aligned}$$

At first, all objects $\breve{a}_j$ will be moved to the environment and the objects $\hat{a}, \#_1, I_c$ to region 2 (thus, we stop without continuing the loop) and after that all objects $\hat{a}_j, b_j, d_j$ will be moved from region 2 to region 1. Hence, in region 2 now there are only the objects c_1 (representing the result of the computation) and the six additional objects $\#_1$, $\#_2$, $\hat{a}$, I_c, q_f, h_{n+4}. □

Both constructions from Theorem 5.6.1 and Theorem 5.6.2 can easily be modified to show that

$$PsOP_2(sym_1, anti_1)_T = PsRE \text{ and}$$
$$PsOP_2(sym_2)_T = PsRE,$$

i.e., the results proved in Theorem 5.6.1 and Theorem 5.6.2 can be extended from sets of natural numbers to sets of vectors of natural numbers.

5.7 One Membrane

In this section we show that systems with symport / antiport of weight one having only one membrane/cell are not very powerful. A similar result concerning systems with symport of weight two may be found in [100].

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

5.7.1 Upper Bound

Theorem 5.7.1 $NOP_1(sym_1, anti_1) \cup NOtP_1(sym_1, anti_1) \subseteq NFIN$.

Proof. Consider an arbitrary P system Π with one membrane and symport / antiport rules of weight 1, $\Pi = (O, E, [_1 \ ]_1, w, R)$. Consider also an arbitrary halting computation, ending in some configuration C with $w_1 \in (O - E)^*$ and $w_e \in E^*$ in membrane 1 and $w_0 \in (O - E)^*$ in the environment. We are claiming that $|w_1| + |w_e| \leq |w|$.

We shall prove this assertion by contradiction. Let us assume the contrary. Since the number of objects in the membrane can only increase by symport rules, some rule $p_0 : (s_0, in)$ had to be applied at some step (by definition $s_0 \in O - E$). This implies that s_0 has been brought to the environment. We can assume that rules $p_i : (s_i, out; s_{i-1}, in)$, $1 \leq i < n$ have been applied ($n \geq 0$), $s_i \in O - E$, $1 \leq i \leq n$. Suppose also that n is maximal (s_n was not brought to the environment by antiport with another object from $O - E$). Thus R contains either a rule $p : (s_n, out)$, or $p' : (s_n, out, a, in)$, $a \in O$.

Now let us examine the final configuration. If s_0 is in w_1, then p_0 can be applied, hence the configuration is not final. Therefore s_0 is in w_0. For all $1 \leq i \leq n$, given s_{i-1} in w_0, if s_i is in w_1, then p_i can be applied, hence the configuration is not final. Consequently, s_i is in w_0 as well. By induction, we obtain that s_n is in w_0. However, this implies that either $p \in R$ and p can be applied, or some $p' \in R$ and p' can be applied, therefore the configuration is not final. This implies that any computation where number of objects inside the membrane is increased cannot halt. Therefore, Π can only generate numbers not exceeding $|w|$. The statement of the theorem follows directly from here. □

5.7.2 Lower Bound

At any step of the computation, each copy of object can be in one of the two possible regions: in region 1 or in the environment. This is a kind of "1-bit memory", and the only way this memory can influence the following computation is by a cooperative transport rule: moving this object to the other region together with moving another object in the same or opposite direction. Therefore, it is expected that the number set generated by such

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

P systems is "continuous", i.e., the distance between any two neighboring numbers is bounded.

More formally, let us use the following notion: the *segments* (finite segments of arithmetic progression with difference k) are defined as $SEG_k = \{\{n+ki \mid 0 \leq i \leq m\} \mid n, m \geq 0\} \cup \{\emptyset\}$. For instance, SEG_1 is the class of all finite sets of consecutive numbers, while SEG_2 is the class of all finite sets of consecutive even numbers and all finite sets of consecutive odd numbers.

Example 5.7.1 $\emptyset \in NOP_1(sym_1, anti_1) \cap NOP_1(sym_2)$.

Consider a P system $\Pi_0 = (O = \{b\}, E = \emptyset, \mu = [_1 \ ]_1, w_1 = b, R = \{(b, in), (b, out)\}, i_0 = 1)$. There is one possible computation: object b oscillates between region 1 and the environment, so the set of results of the halting computations is empty. This system only uses symport rules of weight 1, so it belongs to both $OP_1(sym_1, anti_1)$ and $OP_1(sym_2)$.

Example 5.7.2 $NOP_1(sym_1, anti_1) \supseteq SEG_1$.

Fix the numbers $m, n \geq 0$. Consider a P system $\Pi_1 = (O = \{a, b\}, E = \{a\}, \mu = [_1 \ ]_1, w_1 = a^n b^m, R = \{(b, out), (b, out; a, in)\}, i_0 = 1)$. Any computation of Π_1 halts in at most 1 step: every object b exit region 1, in exchange for either an object a or for nothing.

This is why the computation halts, with region 1 containing n copies of object a initially present there, and some number i of copies of a that were brought inside. Notice that $0 \leq i \leq m$, and every number is possible. Thus, $N(\Pi_1) = \{n+i \mid 0 \leq i \leq m\}$. Since m, n were chosen arbitrary, together with the previous example we obtain the result we claim: SEG_1 can be generated.

Example 5.7.3 $NOP_1(sym_2) \supseteq SEG_1 \cup SEG_2$.

Fix the numbers $m, n \geq 0$. Consider P systems $\Pi_2 = (O = \{a, b\}, E = \emptyset, \mu = [_1 \ ]_1, w_1 = a^{n+m} b^m, R = \{(b, out), (ab, out)\}, i_0 = 1)$, $\Pi_3 = (O = \{a, b\}, E = \emptyset, \mu = [_1 \ ]_1, w_1 = a^{n+2m} b^{2m}, R = \{(bb, out), (ab, out)\}, i_0 = 1)$. Any computation of Π_2 halts in at most 1 step: every object b exit region 1, together with either an object a or for nothing.

This is why the computation halts, with region 1 containing $n+m$ copies of object a initially present there, except some number j of copies of a that were taken outside. Notice that $0 \leq j \leq m$, and every number is possible.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Substituting $j = m - i$, $0 \leq i \leq m$, we obtain $(n + m) - (m - i) = n + i$. Thus, $N(\Pi_2) = \{n + i \mid 0 \leq i \leq m\}$.

System Π_3 has a similar behavior, except there are $2m$ objects b initially present in region 1, and some number $2i$ of them leave region 1 in pairs, $0 \leq i \leq m$, while each of the others comes into the environment together with an object a. The number of objects a remaining in the system is $(n + 2m) - (2m - 2i) = 2i$. Therefore, $N(\Pi_3) = \{n + 2i \mid 0 \leq i \leq m\}$.

In this way, $SEG_1 \cup SEG_2$ can be generated by systems Π_2 and Π_3 for all possible numbers m, n, together with Π_0.

5.8 Symport of Weight Three

We first improve the result $N_{13}OP_1(sym_3) = N_{13}RE$ from [100]. For the proof, we use the variant of counter machines with conflicting counters and implement the semantics that if two conflicting counters are non-empty at the same time, then the computation is blocked without producing a result.

Theorem 5.8.1 $N_7OP_1(sym_3) = N_7RE$.

Proof. Let L be an arbitrary set from N_7RE and consider a counter automaton $M = (d, Q, q_0, q_f, P, C)$ with conflicting counters generating $L-7$ $(= \{n - 7 \mid n \in L\})$; C is a finite set of pair sets of conflicting counters $\{i, \bar{i}\}$. We construct a P system simulating M:

$$\begin{array}{lcl}
\Pi & = & (O, E, [_1\ \ ]_1, w_1, R_1, 1), \\
O & = & \{x_i \mid 1 \leq i \leq 6\} \cup Q \cup \{(p, j) \mid p \in P, 1 \leq j \leq 6\} \\
 & \cup & \{a_i, A_i \mid i \in C\} \cup \{\#, b, d\}, \\
E & = & \{a_i, A_i \mid i \in C\} \cup \{x_2, x_3, \#\} \\
 & \cup & Q \cup \{(p, j) \mid p \in P, j \in \{2, 4, 5, 6\}\} \\
w_1 & = & l_0 d x_1 x_4 x_5 x_6 \prod_{p \in P} (p, 1)\,(p, 3)\, b.
\end{array}$$

The following rules allow us to simulate the counter automaton M:

The rules $(da_i a_{\bar{i}}, out)$ implement the special semantics of conflicting counters $\{i, \bar{i}\}$ with leading to an infinite computation by applying the rules $(d\#, out)$ and $(d\#, in)$.

The simulation of the instructions of M is initiated by also sending out x_1 in the first step; the rules $(x_1 x_2 x_3, in)$ as well as $(x_2 x_4 x_5, out)$ and $(x_3 x_6, out)$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

then allow us to send out the specific signal variables x_4, x_5, and x_6 which are needed to guide the sequence of rules to be applied.

The instruction $p : (l \to l', i-)$ is simulated by the sequence of rules

$$\begin{array}{l}(l(p,1)x_1, out),\\ ((p,1)x_4(p,2), in),\\ ((p,2)(p,3)a_i, out),\ ((p,2)(p,3)d, out),\\ ((p,3)x_5(p,4), in),\\ ((p,4)(p,5), out),\\ ((p,5)x_6 l', in).\end{array}$$

In case that no symbol a_i is present (which corresponds to the fact that counter i is empty), the rule $((p,2)(p,3)d, out)$ leads to an infinite computation by applying the rules $(d\#, out)$ and $(d\#, in)$. Otherwise, decrementing is successfully accomplished by applying the rule $((p,2)(p,3)a_i, out)$.

The instruction $p : (l \to l', i+)$ is simulated by the sequence of rules

$$\begin{array}{l}(l(p,1)x_1, out),\\ ((p,1)x_4(p,2), in),\\ ((p,2)(p,3)A_i, out),\\ ((p,3)x_5 l', in),\\ (A_i x_6 a_i, in).\end{array}$$

The symbol A_i is sent out to take exactly one symbol a_i in.

A simulation of M by Π terminates with sending out the symbols from $\{(p,1), (p,3) \mid p \in P\} \cup \{A_i \mid i \in C\}$ which were used during the simulation of the instructions of M as soon as the halting label l_h of M appears:

$(l_h b x, out),\ x \in \{(p,1), (p,3) \mid p \in P\} \cup \{A_i \mid i \in C\}$,

$(l_h b, in)$.

If the system halts, the objects inside correspond with the contents of the output registers, and the extra symbols are $l_h, d, b, x_1, x_4, x_5, x_6$, i.e., seven in total. □

5.9 Concluding Remarks

We now finish our overview with repeating (some of) the best known results of computational completeness. Results (5.1) were obtained in [96] and [55], while result (5.2) was obtained in [92], [101], [95]. The other proofs, obtained in [24], [31], [22], [30] and [21], are presented in this chapter.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

One membrane

$$PsRE = DPs_aOP_1(anti_2) = DPs_aOP_1(sym_3), \quad (5.1)$$
$$N_1RE = N_1OP_1(anti_2), \quad (5.2)$$
$$N_7RE = N_7OP_1(sym_3). \quad (5.3)$$

P systems - minimal cooperation

$$PsRE = PsOP_2(sym_1, anti_1)_T = PsOP_2(sym_2)_T, \quad (5.4)$$
$$N_3RE = N_3OP_2(sym_1, anti_1), \quad (5.5)$$
$$N_6RE = N_6OP_2(sym_2). \quad (5.6)$$

Tissue P systems - minimal cooperation

$$PsRE = DPs_aOtP_2(sym_1, anti_1) = DPs_aOtP_2(sym_2), \quad (5.7)$$
$$PsRE = PsOtP_2(sym_1, anti_1) = PsOtP_2(sym_2). \quad (5.8)$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 6

Small Number of Objects

6.1 Introduction

P systems. A quite surprising result was presented in [165]: using symport / antiport rules of unbounded weight, P systems with four membranes are computationally complete even when the alphabet contains only three symbols:

$$NRE = NO_3P_4(sym_*, anti_*).$$

Then it has been shown in [13] that

$$NRE = NO_5P_1(sym_*, anti_*),$$

i.e., for P systems with one membrane, even five objects are enough for getting computational completeness.

The original result was improved in [16]; in sum, the actual computational completeness results for P systems can be found there:

$$NRE = NO_nP_m(sym_*, anti_*) = N_aO_nP_m(sym_*, anti_*)$$
$$\text{for } (n,m) \in \{(5,1),(4,2),(3,3),(2,4)\}.$$

The results mentioned above are presented as part of a general picture ("complexity carpet"), including results for generating/ accepting/ computing functions on vectors of specified dimensions.

Below computational completeness. The same article ([16]) presents undecidability results for the families

$$NO_2P_3(sym_*, anti_*),\ NO_3P_2(sym_*, anti_*),\ NO_4P_1(sym_*, anti_*),$$
$$N_aO_2P_3(sym_*, anti_*),\ N_aO_3P_2(sym_*, anti_*),\ N_aO_4P_1(sym_*, anti_*);$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

moreover, it was shown that

$$NO_1P_2(sym_*, anti_*) \cap NO_2P_1(sym_*, anti_*) \supseteq NREG;$$
$$N_aO_3P_1(sym_*, anti_*) \cap N_aO_2P_2(sym_*, anti_*) \supseteq NREG;$$
$$NO_1P_1(sym_*, anti_*) = NFIN;$$
$$N_aO_2P_1(sym_*, anti_*) \supseteq NFIN.$$

The last result has been improved in [121]; in the same article one also presents some results on one-symbol P systems:

$$N_aO_2P_1(sym_*, anti_*) \supsetneq NREG;$$
$$N_aO_1P_{5m+3}(sym_*, anti_*) \supsetneq aPBLIND(m);$$
$$NO_1P_{5m+3}(sym_*, anti_*) \supseteq PBLIND(m).$$

The last two results have been improved in the final version of [121]: instead of $5m + 3$ membranes, $2m + 3$ membranes are enough to simulate partially blind counter automata (acceptors or generators).

Several questions are still open; the most interesting one is to determine the computational power of P systems with one symbol (we conjecture that they are not computationally complete, even if we can use an unbounded number of membranes and symport / antiport rules of unbounded weight).

Tissue P Systems. The question concerning systems with only one object has been answered in a positive way in [94] for tissue P systems:

$$NRE = NO_1tP_7(sym_*, anti_*) = NO_1t'P_6(sym_*, anti_*).$$

In [17] the "complexity carpet" for tissue P systems was completed:

$$NRE = NO_ntP_m(sym_*, anti_*)$$
$$\text{for } (n, m) \in \{(4, 2), (2, 3), (1, 7)\},$$

but

$$NREG = NO_*tP_1(sym_*, anti_*) = NO_2tP_1(sym_*, anti_*)$$

and

$$NFIN = NO_1tP_1(sym_*, anti_*) = NO_1t'P_1(sym_*, anti_*).$$

Using two channels between a cell and the environment, one cell can sometimes be saved, and one-cell systems become computationally complete:

$$NRE = NO_nt'P_m(sym_*, anti_*)$$
$$\text{for } (n, m) \in \{(5, 1), (3, 2), (2, 3), (1, 6)\}.$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

6.2 Definitions

The definitions of P systems and tissue P systems with symport / antiport rules were given in the previous chapter.

In the notations

$$\mathbf{X}O_nP_m(sym_*, anti_*),\ \mathbf{X}O_ntP_m(sym_*, anti_*)$$

for $\mathbf{X} \in \{N, Ps, N_a, Ps_a\}$, we will omit $(sym_*, anti_*)$ for conciseness. Moreover, in case we restrict the vectors to k-dimensional ones, we will replace Ps by $Ps(k)$. We should also mention that we allow the input or output of the system to be distributed (represented by symbols in different regions).

Yet another deviation from the standard notations in this chapter is the following: whenever the set E of environment symbols is omitted, $E = O$, the set of all symbols, is assumed.

Throughout this chapter we will use the notation of symport / antiport rules for tissue P systems, also in the case of P systems (as if there was a channel $(i, parent(i))$ for each membrane i; $parent(i)$ stands for the region immediately outside region i): we will write $u/v \in R_i$ instead of $(u, out; v, in) \in R_i$. Similarly, u/λ stands for (u, out) and λ/v stands for (v, in). We will do it here mainly for conciseness (in Chapter 5 we used a different notation, in particular, in order not to denote symport like a degenerate case of antiport).

6.3 Membrane Case

We now establish our results for P systems with symport / antiport rules and small numbers of membranes and symbols. The main constructions show that a P system with symport / antiport rules and $m \geq 1$ membranes as well as $s \geq 2$ symbols can simulate a register machine with $\max\{m(s-2), (m-1)(s-1)\}$ registers. For example, in that way we improve the result $NRE = NO_3P_4$ as established in [165] to $NRE = NO_3P_3 = NO_2P_4$.

6.3.1 At Least Three Symbols

It was already shown in [13] that any d-register machine can be simulated by a P system in one membrane using $d + 2$ symbols. In this subsection,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

following [16] we generalize this result: P systems with m membranes and $s \geq 3$ symbols can simulate $m(s-2)$-register machines:

Theorem 6.3.1 *Any mn-register machine can be simulated by a P system with $2+n$ symbols and m membranes.*

Proof. Let us consider a register machine $M = (d, R, l_1, l_{halt})$ with $d = mn$ registers. No matter what the goal of M is (generating / accepting vectors of natural numbers, computing functions), we can construct the P system (of degree m)

$$\begin{aligned}
\Pi &= (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m), \\
O &= \{p, q\} \cup \{a_j \mid 1 \leq j \leq n\}, \\
\mu &= [_1 \, [_2 \,]_2 \cdots [_m \,]_m \,]_1, \\
w_1 &= w_0 \prod_{j=1}^{n} a_j^{r_j}, \\
w_i &= \prod_{j=1}^{n} a_j^{r_{j+(i-1)n}}, \; 2 \leq i \leq m,
\end{aligned}$$

that simulates the actions of M as follows. The symbols p and q are needed for encoding the instructions of M; q also has the function of a trap symbol, i.e., in case of the wrong choice for a rule to be applied we take in so many symbols q that we can never again rid of them and therefore get "trapped" in an infinite loop. Throughout the computation, the value of register $j+(i-1)n$ is represented by the multiplicity of symbol a_j in region i. In the generating case, $w_1 = w_0$ and $w_i = \lambda$ for $2 \leq i \leq m$; in the accepting case and in the case of computing functions, the numbers of symbols a_j as defined above specify the input.

An important part of the proof is to define a suitable encoding $c : \mathbb{N} \to \mathbb{N}$ (a strictly monotone linear function) for the instructions of the register machine: As we will use at most 6 different subsequent labels for each instruction, without loss of generality we assume the labels of M to be positive integers such that the labels assigned to ADD and SUB instructions have the values $6i-5$ for $1 \leq i < t$, as well as $l_0 = 1$ and $l_{halt} = 6(t-1)+1$, for some $t \geq 1$.

For the operations assigned to a label l and working on register r, we will use specific encodings by the symbols p and q which allow us to distinguish between the operations ADD, SUBTRACT, and ZERO TEST. As we have d registers, this yields $3d$ multisets for specifying operations. The number of symbols p and q in these operation multisets is taken in such a way that

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

the number of symbols p always exceeds the number of symbols q. Finally, the number of symbols q can never be split into two parts that could be interpreted as belonging to two operation multisets.

Hence, the range for the number of symbols q is taken as the interval $[3d+1, 6d]$ and the range for the number of symbols p is taken as the interval $[6d+1, 9d+1]$. Thus, with $h = 12d+1$ we define the following operation multisets:

$$\begin{array}{ll} \text{ADD:} & \alpha_+(r) = q^{3d+r}p^{h-(3d+r)},\ 1 \leq r \leq d, \\ \text{SUBTRACT:} & \alpha_-(r) = q^{4d+r}p^{h-(4d+r)},\ 1 \leq r \leq d, \\ \text{ZEROTEST:} & \alpha_0(r) = q^{5d+r}p^{h-(5d+r)},\ 1 \leq r \leq d. \end{array}$$

The encoding $c : \mathbb{N} \to \mathbb{N}$ which shall encode the instruction l of M to be simulated as $p^{c(l)}$ also has to obey to the following conditions:

- For any i, j with $1 \leq i, j \leq 6t-5$, $c(i) + c(j) > c(6t-4)$, i.e., the sum of the codes of two instruction labels has to be larger than the largest code we will ever use for the given M, hence, if we do not use the maximal number of symbols p as interpretation of a code for an instruction (label), then the remaining rest of symbols p cannot be misinterpreted as the code for another instruction label.
- The distance g between any two codes $c(i)$ and $c(i+1)$ has to be larger than any of the multiplicities of the symbol p which appear besides codes in the rules defined above.

As we shall see in the construction of the rules below, we may take

$$g = 2h = 24d + 2.$$

In sum, for a function c fulfilling all the conditions stated above we can take

$$c(x) = g(x + 6t - 4) \text{ for } x \geq 0.$$

For example, with this function, for arbitrary $i, j \geq 1$ we get

$$c(i)+c(j) = g(i+6t-4)+g(j+6t-4) > g(6t-4+6t-4) = c(6t-4).$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Moreover, for $l_1 = 1$ we therefore obtain

$$c\,(l_1) = g\,(6t-3) = (24d+2)\,(6t-3)$$

as well as

$$w_0 = p^{c(l_1)} = p^{(24d+2)(6t-3)}.$$

Finally, we have to find a number f which is so large that after getting f symbols we inevitably enter an infinite loop with the rule

$$q^f/q^{3f};$$

as we shall justify below, we can take

$$f = c\,(l_{halt}+1) = 2g(6t-4).$$

Equipped with this coding function and the constants defined above we are now able to define the following set of symport / antiport rules assigned to the membranes for simulating the actions of the given register machine M:

$$\begin{aligned}
R_1 \;=\;& \{p^{c(l_1)}/p^{c(l_2)}a_s, p^{c(l_1)}/p^{c(l_3)}a_s \mid \\
& l_1 : (A(s), l_2, l_3) \in R, 1 \le s \le n\} \\
\cup\;& \{p^{c(l_1)}/p^{c(l_1+1)}\alpha_+(s+(s'-1)n)a_s, p^{c(l_1+1)}/p^{c(l_1+2)}, \\
& p^{c(l_1+2)}/p^{c(l_1+3)}, p^{c(l_1+3)}\alpha_+(s+(s'-1)n)/p^{c(l_2)} \\
& p^{c(l_1+3)}\alpha_+(s+(s'-1)n)/p^{c(l_3)} \mid \\
& l_1 : (A(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 2 \le s' \le m\} \\
\cup\;& \{p^{c(l_1)}a_s/p^{c(l_2)}, p^{c(l_1)}/p^{c(l_1+1)}\alpha_0(s), \\
& p^{c(l_1+1)}/p^{c(l_1+2)}, \alpha_0(s)a_s/q^{3f}, \\
& p^{c(l_1+2)}\alpha_0(s)/p^{c(l_3)} \mid l_1 : (S(s), l_2, l_3) \in R, 1 \le s \le n\} \\
\cup\;& \{p^{c(l_1)}/p^{c(l_1+1)}\alpha_-(s+(s'-1)n), p^{c(l_1+1)}/p^{c(l_1+2)}, \\
& p^{c(l_1+2)}/p^{c(l_1+3)}, p^{c(l_1+3)}\alpha_-(s+(s'-1)n)a_s/p^{c(l_2)}, \\
& p^{c(l_1)}/p^{c(l_1+4)}\alpha_0(s+(s'-1)n), p^{c(l_1+4)}/p^{c(l_1+5)}, \\
& p^{c(l_1+5)}\alpha_0(s+(s'-1)n)/p^{c(l_3)}, \\
& \alpha_-(s+(s'-1)n)/q^{3f} \mid \\
& l_1 : (S(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 2 \le s' \le m\} \\
\cup\;& \{p^{c(l_{halt})}/\lambda, p^h/q^{3f}, q^f/q^{3f}\}
\end{aligned}$$

as well as for $2 \le s' \le m$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned} R_{s'} &= \{\lambda/\alpha_+(s+(s'-1)n)a_s, \alpha_+(s+(s'-1)n)/\lambda \mid \\ &\quad l_1 : (A(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 2 \le s' \le m\} \\ &\cup \{a_s/\alpha_-(s+(s'-1)n), \alpha_-(s+(s'-1)n)/\lambda, \\ &\quad a_s/\alpha_0(s+(s'-1)n) \mid \\ &\quad l_1 : (S(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 2 \le s' \le m\} \end{aligned}$$

The correct work of the rules can be described as follows:

1. Throughout the whole computation in Π, it is directed by the code $p^{c(l)}$ for some $l \le 6t-5$; in order to guarantee the correct sequence of encoded rules the trap is activated in case of a wrong choice, which in any case guarantees an infinite loop with the symbols q by the "trap rule"

   $$q^f/q^{3f}.$$

   The minimal number of superfluous symbols p to start the trap is h and causes the application of the rule p^h/q^{3f}.

2. For each ADD instruction $l_1 : (A(s), l_2, l_3)$ of M, i.e., for incrementing register s for $1 \le s \le n$, we use the following rules in R_1:

   $$p^{c(l_1)}/p^{c(l_2)}a_s, \text{ and}$$
   $$p^{c(l_1)}/p^{c(l_3)}a_s.$$

   In that way, the ADD instruction $l_1 : (A(s), l_2, l_3)$ of M for one of the first n registers is simulated in only one step: the number of symbols p representing the instruction of M labelled by l_1 is replaced by the number of symbols p representing the instruction of M labelled by l_2 or l_3, respectively, in the same moment also incrementing the number of symbols a_s. Whenever a wrong number of symbols p is taken, the remaining symbols cannot be used by another rule than the "trap rule" p^h/q^{3f}, which in the succeeding computation steps inevitably leads to the repeated application of the rule q^f/q^{3f} thus flooding the skin membrane with more and more symbols q.
   On the other hand, incrementing register $s+(s'-1)n$, for $1 \le s \le n$, $2 \le s' \le m$, i.e., registers $n+1$ to nm is accomplished by the rules

   $$p^{c(l_1)}/p^{c(l_1+1)}\alpha_+(s+(s'-1)n)a_s,$$
   $$p^{c(l_1+1)}/p^{c(l_1+2)}$$
   $$p^{c(l_1+2)}/p^{c(l_1+3)},$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$p^{c(l_1+3)}\alpha_+(s+(s'-1)n)/p^{c(l_2)}$
$p^{c(l_1+3)}\alpha_+(s+(s'-1)n)/p^{c(l_3)}$

in R_1 as well as by the rules

$\lambda/\alpha_+(s+(s'-1)n)a_s$,
$\alpha_+(s+(s'-1)n)/\lambda$ in $R_{s'}$

Hence, adding one to the contents of registers $n+1$ to nm now needs four steps: the number of symbols p representing the instruction of M labelled by l_1 is replaced by $p^{c(l_1+1)}$ together with $3d+s+(s'-1)n$ additional symbols q, $h-(3d+s+(s'-1)n)$ symbols p and the symbol a_s. In the second step, $p^{c(l_1+1)}$ is exchanged with $p^{c(l_1+2)}$, while at the same time the additional $3d+s+(s'-1)n$ symbols q and $h-(3d+s+(s'-1)n)$ symbols p are introduced together with a_s in membrane s'. In the third step, the $c(l_1+2)$ symbols p in the skin membrane are exchanged with $c(l_1+2)$ symbols p from the environment, whereas the additional $3d+s+(s'-1)n$ symbols q and $h-(3d+s+(s'-1)n)$ symbols p pass out from membrane r. Finally, in the fourth step, these latter symbols together with $p^{c(l_1+3)}$ in the skin membrane are replaced by the number of symbols p representing the next instruction of M labelled by l_2 or l_3, respectively.

3. For simulating the decrementing step of a SUB instruction $l_1 : (S(s), l_2, l_3)$ from R we introduce the following rules:

   $p^{c(l_1)}a_s/p^{c(l_2)}$

   for decrementing the contents of register s, for $1 \leq s \leq n$, represented by the symbols a_s in the skin membrane.
   In that way, the decrementing step of the SUB instruction $l_1 : (S(s), l_2, l_3)$ of M now is also simulated in one step: together with $p^{c(l_1)}$ we send out one symbol a_s and take in $p^{c(l_2)}$, which encodes the label of the instruction that has to be executed after the successful decrementing of register s, for $1 \leq s \leq n$.
   For decrementing the registers $s+(s'-1)n$, for $1 \leq s \leq n$, $2 \leq s' \leq m$, we need the following rules:

   $p^{c(l_1)}/p^{c(l_1+1)}\alpha_-(s+(s'-1)n)$,
   $p^{c(l_1+1)}/p^{c(l_1+2)}$
   $p^{c(l_1+2)}/p^{c(l_1+3)}$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$p^{c(l_1+3)}\alpha_-(s+(s'-1)n)a_s/p^{c(l_2)} \text{ in } R_1$$

as well as

$$a_s/\alpha_-(s+(s'-1)n),$$
$$\alpha_-(s+(s'-1)n)/\lambda \text{ in } R_r.$$

In this case, the SUB instruction is simulated in four steps: $p^{c(l_1)}$ is replaced by $p^{c(l_1+1)}$ together with the "operation multiset" $\alpha_-(s+(s'-1)n)$, i.e., $q^{4d+r}p^{h-(4d+r)}$, $r = s+(s'-1)n$, for $1 \leq s \leq n$, $2 \leq s' \leq m$. While in the next two steps, two intermediate exchanges of symbols p with the environment take place, the symbol a_s is exchanged with $\alpha_-(s+(s'-1)n)$ in membrane r, that, in the third step, goes out again to the skin membrane, where it can now together with $p^{c(l_1+3)}$ be exchanged with $p^{c(l_2)}$, i.e., the representation of the next instruction of M.
Again we notice that if we do not choose the correct rule, then the trap is activated by the rule p^h/q^{3f}, especially if no symbol a_s is present in membrane r, then we have to apply the "trap rule" $\alpha_-(s+(s'-1)n)/q^{3f}$.

4. For simulating the zero test, i.e., the case where we check the contents of register r to be zero, of a SUB instruction $l_1 : (S(s), l_2, l_3)$ from R for registers 1 to n we take the following rules:

$$p^{c(l_1)}/p^{c(l_1+1)}\alpha_0(s),$$
$$p^{c(l_1+1)}/p^{c(l_1+2)}, \text{ and}$$
$$p^{c(l_1+2)}\alpha_0(s)/p^{c(l_3)} \text{ in } R_1.$$

If the rule $\alpha_0(s)a_s/q^{3f}$ from R_1 can be applied, then in the next step we cannot apply $p^{c(l_1+2)}\alpha_0(s)/p^{c(l_3)}$ from R_1, hence, only a rule using less than $c(l_1+2)$ symbols p can be used together with the "trap rule" p^h/q^{3f}.

For simulating the zero test, i.e., the case where we check the contents of register r to be zero, of a SUB instruction $l_1 : (S(s), l_2, l_3)$ from R for registers $n+1$ to nm we now take the following rules:

$$p^{c(l_1)}/p^{c(l_1+4)}\alpha_0(s+(s'-1)n),$$
$$p^{c(l_1+4)}/p^{c(l_1+5)}, \text{ and}$$
$$p^{c(l_1+5)}\alpha_0(s+(s'-1)n)/p^{c(l_3)} \text{ in } R_1.$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

If the rule $a_s/\alpha_0(s+(s'-1)n)$ from R_r can be applied, then in the next step we cannot apply $p^{c(l_1+5)}\alpha_0(s+(s'-1)n)/p^{c(l_3)}$ from R_1, hence, only a rule using less than $c(l_1+5)$ symbols p can be used together with the "trap rule" p^h/q^{3f}.

5. The number of symbols p never exceeds $c\,(l_{halt}) = 2g(6t-4)$ as long as the simulation of instructions from R works correctly. By definition, $f = c\,(l_{halt}+1) = 2g(6t-4)$, hence, there will be at least three times more symbols q in region 1 than symbols p in the system after having applied a "trap rule", thus introducing $3f$ symbols q. As by any rule in R_1, the number of symbols p coming in is less than double the number sent out, the total number of symbols p in the system, in one computation step, can at most be doubled in total, too. As every rule that removes some symbols q from region 1 involves at least as many symbols p as symbols q, the "trap rule" q^f/q^{3f} guarantees that in the succeeding steps this relation will still hold true, no matter how the present symbols p and q are interpreted for rules in Π. Therefore, if as soon as a "trap rule" has been applied, then the number of objects q will grow and the system will never halt.

6. Finally, for the halt label $l_{halt} = 6t-5$ we only take the rule

   $$p^{c(l_{halt})}/\lambda,$$

   hence, the work of Π will stop exactly when the work of M stops (provided the trap has not been activated due to a wrong non-deterministic choice during the computation).

From the explanations given above we conclude that Π halts if and only if M halts, and moreover, the final configuration of Π represents the final contents of the registers in M. These observations conclude the proof. □

As already proved in [13], when using P systems with only one membrane, at most five objects are needed to obtain computational completeness:

Corollary 6.3.1 $NO_5P_1 = N_aO_5P_1 = NRE$.

Moreover, from Theorem 6.3.1 we can also conclude that P systems with two membranes are computationally complete with only four objects:

Corollary 6.3.2 $NO_4P_2 = N_aO_4P_2 = NRE$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

6.3.2 At Least Two Symbols and at Least Two Membranes

On the other hand, for $s, m \geq 2$, we can show that P systems with s symbols and m membranes can simulate $(s-1)(m-1)$-register machines:

Theorem 6.3.2 *Any mn-register machine can be simulated by a P system with $n+1$ symbols and $m+1$ membranes, with $n, m \geq 1$.*

Proof. Consider a register machine $M = (d, R, l_0, l_{halt})$ with $d = mn$ registers. We construct the P system

$$\begin{array}{lcl} \Pi & = & (O, \mu, w_1, \cdots, w_{m+1}, R_1, \cdots, R_{m+1}), \\ O & = & \{p\} \cup \{a_j \mid 1 \leq j \leq n\}, \\ \mu & = & [_1 \, [_2 \,]_2 \cdots [_{m+1} \,]_{m+1} \,]_1, \\ w_1 & = & w_0, \\ w_{i+1} & = & \prod_{j=1}^{n} a_j^{r_{j+(i-1)n}}, \ 1 \leq i \leq m, \end{array}$$

that simulates the actions of M as follows. The contents of register $j+(i-1)n$ is represented by the multiplicity of symbols a_j in region $i+1$, whereas the symbol p is needed for encoding the instructions of M; this time, too many copies of a_1 in the skin membrane have the function of trap symbols.

Again, an important part of the proof is to define a suitable encoding $c : \mathbb{N} \to \mathbb{N}$ for the instructions of the register machine, and at most 6 different subsequent labels will be used for each instruction, hence, without loss of generality we assume the labels of M to be positive integers such that the labels assigned to ADD and SUB instructions have the values $6i-5$ for $1 \leq i < t$, as well as $l_0 = 1$ and $l_{halt} = 6(t-1)+1$, for some $t \geq 1$.

Since one copy of a_s will be used for addition/subtraction, now the operation multisets will be encoded by even numbers of object a_1, i.e., we take $h = 2(12d+1) = 24d+2$ and define the following operation multisets:

$$\begin{array}{ll} \text{ADD:} & \alpha_+(r) = {a_1}^{6d+2r} p^{h-(6d+2r)}, \ 1 \leq r \leq d, \\ \text{SUBTRACT:} & \alpha_-(r) = {a_1}^{8d+2r} p^{h-(8d+2r)}, \ 1 \leq r \leq d, \\ \text{ZEROTEST:} & \alpha_0(r) = {a_1}^{10d+2r} p^{h-(10d+2r)}, \ 1 \leq r \leq d. \end{array}$$

In a similar way as before, we now take

$$g = 2h = 48d+4$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

and define the function c by

$$c\,(x) = g(x + 6t - 4) \text{ for } x \geq 0.$$

For $l_1 = 1$ we therefore obtain

$$c\,(l_1) = g\,(6t - 3) = (48d + 4)\,(6t - 3)$$

as well as

$$w_0 = p^{c(l_1)} = p^{(48d+4)(6t-3)}.$$

Finally, for f we again take

$$f = c\,(l_{halt} + 1) = 2g(6t - 4)$$

which is so large that after getting f symbols we inevitably enter an infinite loop with the rule

$${a_1}^f / {a_1}^{3f}.$$

Equipped with this coding function and the constants defined above we are now able to define the following set of symport / antiport rules assigned to the membranes for simulating the actions of the given register machine M:

$$\begin{aligned}
R_1 \;=\;& \{p^{c(l_1)}/p^{c(l_{1+1})}\alpha_+(s + (s' - 1)n)a_s, p^{c(l_1+1)}/p^{c(l_1+2)},\\
& p^{c(l_1+2)}/p^{c(l_1+3)}, p^{c(l_1+3)}\alpha_+(s + (s' - 1)n)/p^{c(l_2)},\\
& p^{c(l_1+3)}\alpha_+(s + (s' - 1)n)/p^{c(l_3)} \mid\\
& l_1 : (A(s + (s' - 1)n), l_2, l_3), 1 \leq s \leq n, 1 \leq s' \leq m\}\\
\cup\;& \{p^{c(l_1)}/p^{c(l_1+1)}\alpha_-(s + (s' - 1)n), p^{c(l_1+1)}/p^{c(l_1+2)},\\
& p^{c(l_1+2)}/p^{c(l_1+3)}, p^{c(l_1+3)}\alpha_-(s + (s' - 1)n)a_s/p^{c(l_2)},\\
& \alpha_-(s + (s' - 1)n)/{a_1}^{3f},\\
& p^{c(l_1)}/p^{c(l_1+4)}\alpha_0(s + (s' - 1)n), p^{c(l_1+4)}/p^{c(l_1+5)},\\
& p^{c(l_1+5)}\alpha_0(s + (s' - 1)n)/p^{c(l_3)} \mid\\
& l_1 : (S(s + (s' - 1)n), l_2, l_3), 1 \leq s \leq n, 1 \leq s' \leq m\}\\
\cup\;& \left\{p^{c(l_{halt})}/\lambda, p^h/{a_1}^{3f}, {a_1}^f/{a_1}^{3f}\right\}
\end{aligned}$$

and for $1 \leq r \leq m$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned} R_{r+1} &= \{\lambda/\alpha_+(s+(s'-1)n)a_r, \alpha_+(s+(s'-1)n)/\lambda \mid \\ &\quad l_1 : (A(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 1 \le s' \le m\} \\ &\cup \{a_s/\alpha_-(s+(s'-1)n), \alpha_-(s+(s'-1)n)/\lambda, \\ &\quad a_s/\alpha_0(s+(s'-1)n) \mid \\ &\quad l_1 : (S(s+(s'-1)n), l_2, l_3), 1 \le s \le n, 1 \le s' \le m\}. \end{aligned}$$

The operations ADD, SUBTRACT, and ZERO TEST now are carried out for all registers r as in the preceding proof for the registers $r > n$. Hence, we do not repeat all the arguments of the preceding proof, but stress the following important differences:

We now take advantage that the operation multisets additionally satisfy the property that now the number of symbols p can never be split into two parts that could be interpreted as belonging to two operation multisets; this guarantees that during a correct simulation, inside an elementary membrane at most one operation can be executed - and if it is the wrong one (i.e., we do not use all symbols p, but instead use more symbols a_1 from the amount representing the contents of a register), then we return a number of symbols p which is too small to allow the correct rule to be applied from R_1, instead the "trap rule" $p^h/{a_1}^{3f}$ will be applied. □

From the result proved above we can immediately conclude the following, thus also improving the result from [165] where $NO_3P_4 = NRE$ was proved: we can reduce the number of membranes from four to three when using only three objects or the number of symbols from three to two when using four membranes.

Corollary 6.3.3 $NRE = NO_3P_3 = N_aO_3P_3 = NO_2P_4 = N_aO_2P_4$.

6.3.3 One Membrane

The following characterization of $NFIN$ by P systems with only one membrane and only one symbol corresponds with the similar characterization by tissue P systems with only one cell and only one symbol as established in [17].

Example 6.3.1 $NO_1P_1 = NFIN$.

Consider an arbitrary non-empty set $M \in NFIN$. Then we construct a P system $\Pi = (\{a\}, [_1\ ]_1, w_1, R_1)$ where $w_1 = a^m$ with $m = \max(M) + 1$ and $R_1 = \{a^m/a^j \mid j \in M\}$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Clearly, $j < m$ for any $j \in M$, so the computation finishes in one step generating the elements of M as the corresponding number of symbols a in the skin membrane.

The special case of generating the empty set can be done by the following trivial P system: $\Pi = (\{a\}, [_1\ ]_1, a, \{a/a\})$. A computation in this system will never halt.

The inclusion $NFIN \supseteq NO_1P_1$ can easily be argued (like in [17]) as follows:

Consider a P system $\Pi = (\{a\}, [_1\ ]_1, w_1, R_1)$.

Let $m = \min\{j \mid j/i \in R_1 \text{ for some } i\}$. Then a rule from R_1 can be applied as long as region 1 contains at least m objects. Therefore, $N(\Pi) \subseteq \{j \mid j < m\}$; hence, $N(\Pi) \in NFIN$.

Let us recall another relatively simple construction for tissue P systems from [17] that also shows a corresponding result for the membrane case.

Example 6.3.2 *$NO_2P_1 \supseteq NREG$.*

We will use the fact that for any regular set M of non-negative integers there exist finite sets of numbers M_0, M_1 and a number k such that $M = M_0 \cup \{i + jk \mid i \in M_1, j \in \mathbb{N}\}$ (this follows, e.g., from the shape of the minimal finite automaton accepting the unary language with length set M).

We now construct a P system $\Pi = (\{a, p\}, [_1\ ]_1, w_1, R_1)$ where $w_1 = pp$ and $R_1 = \{pp/a^i \mid i \in M_0\} \cup \{pp/pa, pa/pa^{k+1}\} \cup \{pa/a^i \mid i \in M_1\}$, which generates M as the number of symbols a in the skin membrane in halting computations.

Initially, there are no objects a in region 1, so the system "chooses" between generating an element of M_0 in one step or exchanging pp by pa. In the latter case, there is only one copy of p in the system. After an arbitrary number j of applications of the rule pa/pa^{k+1} a rule exchanging pa by a^i for some $i \in M_1$ is eventually applied, generating $jk + i$ symbols a. Hence, $N(\Pi) = M_0 \cup \{i + jk \mid i \in M_1, j \in \mathbb{N}\} = M$.

We will now show two simple constructions to illustrate the accepting power of P systems with one membrane.

Example 6.3.3 *$\{ki \mid i \in \mathbb{N}\} \in N_aO_1P_1$ for any $k \in \mathbb{N}$.*

The set of numbers divisible by a fixed number k (represented by the multiplicity of the object a in the initial configuration) can be accepted by the

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

P system $\Pi = (\{a\}, [_1\ ]_1, w_1, \{a^k/\lambda, a/a\})$*; w_1 is the input of the P system in the initial configuration. The rule a^k/λ sends objects out in groups of k, while the rule a/a "keeps busy" all objects not used by the other one. Hence, the system halts if and only if a multiple of k symbols a has been sent out in several steps finally not using the antiport rule a/a anymore.*

Example 6.3.4 $NFIN \subseteq N_aO_2P_1$.

Any finite set M of natural numbers (represented by the multiplicity of the object a in the initial configuration) can be accepted by the P system $\Pi = (\{a, p\}, [_1\ ]_1, pw_1, \{a/a, p/p\} \cup \{pa^n/\lambda \mid n \in M\})$*; w_1 is the input of the P system in the initial configuration as the number of symbols a in the skin membrane representing the corresponding element from M. The rule pa^n/λ can send out p together with a "correct" number of objects a, while the rules a/a and p/p (in the case of $w_1 = \lambda$) "keep busy" all other objects.*

Example 6.3.3 illustrates that even P systems with one membrane and one object can accept some infinite sets (as opposed to the generating case, where we exactly get all finite sets). Example 6.3.4 shows that when using two objects it is already possible to accept all finite sets.

6.3.4 One Symbol

If only one symbol is available, then so far we do not know whether computational completeness can be obtained even when not bounding the number of membranes (in contrast to tissue P systems which have been shown to be computationally complete with at most seven cells, see [94]). Yet at least we can generate any regular set of natural numbers in only two membranes (remember that with only one membrane we have got a characterization of $NFIN$, see Example 6.3.1).

Example 6.3.5 $NO_1P_2 \supseteq NREG$.

Any finite set can be generated without using the second membrane (see Example 6.3.1), so we proceed with infinite sets. Let $M \in NREG - NFIN$, then there exist finite sets M_0, M_1 with $M_1 \neq \emptyset$ and a number $k > 0$ such that $M = M_0 \cup \{i + jk \mid i \in M_1,\ j \in \mathbb{N}\}$.

Let m be the smallest number in M such that $m > \max(M_0 \cup M_1 \cup \{2k\})$; moreover, let $m' = m + 2k$ (thus, $m' \in M$). Then we consider the P system constructed as follows:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$\Pi = (\{a\}, [_1\ [_2\ ]_2\ ]_1, a^{m'}, \lambda, R_1, R_2)$ *where*
$R_1 = \{a^{m'}/a^i \mid i \in M_0\} \cup \{a^{m'}/a^m, a^m/a^{m+k}\} \cup \{a^m/a^i \mid i \in M_1\}$,
$R_2 = \lambda/a$,
We assume the result of a halting computation to be collected in the second membrane, and we claim $N(\Pi) = M$*:*

$N(\Pi) \supseteq M$*:*

The elements of M_0 *are generated in one step, while the rest of* M *can be generated by*

$[_1\ a^{m'}[_2\ ]_2\ ]_1 \Rightarrow [_1\ a^m[_2\ ]_2\ ]_1 => [_1\ a^{m+k}[_2\ ]_2\ ]_1 \Rightarrow^{j-1}$
$[_1\ a^{m+k}[_2\ a^{(j-1)k}\ ]_2\ ]_1 \Rightarrow [_1\ a^i[_2\ a^{jk}\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ a^{i+jk}\ ]_2\ ]_1$

or by

$[_1\ a^{m'}[_2\ ]_2\ ]_1 \Rightarrow [_1\ a^m[_2\ ]_2\ ]_1 \Rightarrow [_1\ a^i[_2\ ]_2\ ]_1 \Rightarrow [_1\ [_2\ a^i\ ]_2\ ]_1$.

$N(\Pi) \subseteq M$*:*

What other derivations can we get different from those described above?

– *If all* m' *symbols enter membrane* 2, $m' \in M$.

– *If all* m *symbols enter membrane* 2 *(possibly after some additions of* k*),* $m + jk \in M$ *(by the definition of* m, $m \in M$ *and, moreover, it can be prolongued by multiples of* k*).*

– *If during the first step* m *copies of the symbol* a *are used instead of* m' *(and* $2k$ *fall inside), then the system generates some number* $2k + (i + jk)$ *or* $2k + (m + jk)$*; all these numbers belong to* M*, too.*

Nothing else can happen, because $m + k < m'$ *and* $\max(M_0 \cup M_1) < m$ *and because all symbols not used by* R_1 *fall into region* 2.

Just recently, the computational power of P systems with symport / antiport rules and small numbers of symbols and membranes has also been investigated in [121]. There the authors show that P systems with symport / antiport rules with one symbol and three membranes or with two symbols and one membrane can accept the non-semilinear set $L = \{2^n \mid n \geq 0\}$. Moreover, they prove that for any $k \geq 1$, the class of sets of k-tuples of non-negative integers accepted by partially blind (multi-)counter machines is a proper subclass of the class of sets of k-tuples accepted by P systems with symport / antiport rules with one object and multiple membranes. Similarly, the class of sets of k-tuples of non-negative integers generated by partially blind counter machines is shown to be a subclass (but is not known to be proper) of the class of sets of k-tuples generated by P systems with one object and multiple membranes.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Yet the interesting question whether or not P systems with one symbol are universal still remains open.

6.3.5 Summary

From the main theorems (Theorem 6.3.1 and Theorem 6.3.2) of this section showing that P systems with symport / antiport rules and $m \geq 1$ membranes as well as $s \geq 2$ symbols can simulate a register machine with $\max\{m(s-2), (m-1)(s-1)\}$ registers we infer the following general results:

Theorem 6.3.3 *$NO_sP_m = N_aO_sP_m = NRE$, for $m \geq 1$, $s \geq 2$, $m+s \geq 6$.*

We conjecture that these results establishing the computational completeness bounds are optimal.

As the halting problem for d-register machines is undecidable for $d \geq 2$, from Theorems 6.3.1 and 6.3.2 we also obtain the following result:

Theorem 6.3.4 *The halting problem for P systems with symport/ antiport rules and $s \geq 2$ symbols as well as $m \geq 1$ membranes such that $m+s \geq 5$ is undecidable.*

As 1-register machines can generate/ accept all regular number sets, we obtain the following:

Theorem 6.3.5 *$NO_3P_1 \cap NO_2P_2 \cap N_aO_3P_1 \cap N_aO_2P_2 \supseteq NREG$.*

The main results established in this paper now can be summarized in the following table:

In the table depicted above, the class of P systems indicated by

A generates exactly $NFIN$;

B generates at least $NREG$;

C accepts strictly more than $NREG$ ([121], final version) and generates at least $NREG$;

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

	Membranes					
$\|O\|$	1	2	3	4	5	m
1	A	B	B	B	C	D
2	C	1	2 (U)	$\boxed{3}$	4	$m-1$
3	1	**2** (U)	$\boxed{4}$	6	8	$2m-2$
4	2 (U)	$\boxed{4}$	**6**	9	12	$3m-3$
5	$\boxed{3}$	6	9	**12**	16	$4m-4$
6	4	8	12	16	**20**	$5m-5$
s	$s-2$	$2s-4$	$3s-6$	$4s-8$	$5s-10$	$\max\{m(s-2), (m-1)(s-1)\}$

Table 6.1: Classes O_sP_m

D can simulate any partially blind counter automaton ([121]);

d can simulate any d-register machine.

A box around a number indicates a known computational completeness bound, (U) indicates a known unpredictability bound, and a number in boldface shows the diagonal where Theorem 6.3.2 and Theorem 6.3.1 provide the same result (because in that case $m(s-2)$ equals $(m-1)(s-1)$); the numbers above this diagonal are taken from Theorem 6.3.2, while the numbers below the diagonal are taken from Theorem 6.3.1.

Based on these simulation results, we now could discuss in more detail how many symbols s and membranes m at most are needed to accept or generate recursively enumerable sets of vectors of natural numbers or compute functions $\mathbb{N}^k \to \mathbb{N}^l$ (e.g., recursively enumerable sets of d-dimensional vectors, $d \geq 1$, can be generated / accepted by P systems with symport / antiport rules using at most $d+4$ symbols in one membrane, see also [13]). Yet as all these results are direct consequences of the corresponding computational power of the simulated register machines, we do not follow this line any further.

6.4 Tissue Case

We consider tissue P systems with symport / antiport rules and investigate their computational power when using only a (very) small number of symbols

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

and cells. Even when using only one symbol, we need at most six (seven when allowing only one channel between a cell and the environment) cells to generate any recursively enumerable set of natural numbers. On the other hand, with only one cell we can only generate regular sets when using one channel with the environment, whereas one cell with two channels between the cell and the environment obtains computational completeness with at most five symbols. Between these extreme cases of one symbol and one cell, respectively, there seems to be a trade-off between the number of cells and the number of symbols, e.g., for the case of tissue P systems with two channels between a cell and the environment we show that computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively. Moreover, we also show that some variants of tissue P systems characterize the families of finite or regular sets of natural numbers.

6.4.1 One Symbol

In [94] it was shown that one symbol is enough for obtaining computational completeness when using at least seven cells:

Theorem 6.4.1 *$NRE = NO_1tP_n$ for all $n \geq 7$.*

Omitting the condition that for any i, j only one channel out of $\{(i,j),(j,i)\}$ is allowed, at least one cell can be saved (i.e., the one used as the trap, see [94]):

Theorem 6.4.2 *$NRE = NO_1t'P_n$ for all $n \geq 6$.*

If we use a single symbol in only one cell, we exactly get the finite sets:

Example 6.4.1 *To each non-empty finite set L of natural numbers we can construct the tissue P system $\Pi = \left(1, \{a\}, w_1, \{(1,0)\}, R_{(1,0)}\right)$ where $w_1 = a^m$ with $m = \max\{i \mid a^i \in L\} + 1$ and $R_{(1,0)} = \{a^m/a^j \mid a^j \in L\}$. Obviously, $N(\Pi) = L$.*

For the special case of generating the empty set, we can take the following trivial tissue P system $\Pi = (1, \{a\}, a, \{(1,0)\}, \{a/a\})$. A computation in this system will never halt.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Example 6.4.2 *Let* $\Pi = \left(1, \{a\}, a^m, \{(0,1),(1,0)\}, R_{(0,1)}, R_{(1,0)}\right)$ *be a tissue P system with arbitrary sets of symport / antiport rules* $R_{(0,1)}$ *and* $R_{(1,0)}$. *If* $R_{(0,1)} \cup R_{(1,0)}$ *is empty, then obviously* $N(\Pi) = \{m\}$. *Otherwise, let*

$$n = \min\left\{j \mid j/i \in R_{(1,0)} \text{ or } i/j \in R_{(0,1)} \text{ for some } i\right\}.$$

Then a computation in Π *can never halt as long as there are at least* n *objects in the cell. Therefore,* $N(\Pi) \subseteq \{j \mid j < n\}$; *hence,* $N(\Pi) \in NFIN$.

The proof of the following theorem directly follows from the preceding examples:

Theorem 6.4.3 $NFIN = NO_1tP_1 = NO_1t'P_1$.

Proof. In Example 6.4.1, we have shown that $NFIN \subseteq NO_1tP_1$, already by definition, we have the inclusion $NO_1tP_1 \subseteq NO_1t'P_1$, and in Example 6.4.2, the inclusion $NO_1t'P_1 \subseteq NFIN$ has been proved. Altogether, these inclusions prove the statement of the theorem. □

If we use two symbols, we can already generate infinite sets:

Example 6.4.3 *Consider the tissue P system*
$\Pi = \left(2, \{a\}, \lambda, a, \{(2,0),(2,1)\}, R_{(2,0)}, R_{(2,1)}\right)$
with $R_{(2,0)} = \{a^1/a^2\}$ *and* $R_{(2,1)} = \{a^1/\lambda, a^2/\lambda\}$. *The second cell can provide the first one with an arbitrary number of symbols a as long as its contents is not taken as a whole to cell* 1 *by one of the rules from* $R_{(2,1)}$. *Either the computation stops by immediately using the rule* a^1/λ *from* $R_{(2,1)}$ *in the first step or by using* a^2/λ *from* $R_{(2,1)}$ *in one of the succeeding steps. Obviously,* $N(\Pi) = \mathbb{N} - \{0\}$.

If we allow at least three cells or else two channels between a cell and the environment, then at least all regular sets can be generated:

Example 6.4.4 *Let* $G = (\{X_i \mid 1 \leq i \leq n\}, \{a\}, P, X_1)$ *be a regular grammar with productions of the form* $X_i \to aX_j$ *and exactly one production of the form* $X_n \to \lambda$ *(any regular set over a one-letter alphabet can be generated by such a regular grammar); moreover, let* $L(G)$ *denote the language generated by* G.

First consider the tissue P system

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$\Pi' = \left(2, \{a\}, \lambda, a^2, \{(0,2), (2,0), (2,1)\}, R_{(0,2)}, R_{(2,0)}, R_{(2,1)}\right)$
with $R_{(0,2)} = \{a^{2n+2}/a\}$, $R_{(2,0)} = \{a^{2i}/a^{2j+1} \mid X_i \to aX_j \in P\} \cup \{a^{2n}/\lambda\}$, *and* $R_{(2,1)} = \{a/\lambda\}$. *Each non-terminal symbol* X_i *from* N *is encoded by* $2i$ *copies of* a; *in that way, the rules in* $R_{(2,0)}$ *can simulate the productions from* P *in an obvious way; in each computation step in* Π, *also one symbol* a *passes from the second cell to the first one by the rule* a/λ *from* $R_{(2,1)}$. *The "trap rule"* a^{2n+2}/a *from* $R_{(0,2)}$ *guarantees that the simulation works correctly, because if the simulating rule from* $R_{(2,0)}$ *together with the rule from* $R_{(2,1)}$ *does not exhaust the whole contents of the second cell, then the rule from* $R_{(0,2)}$ *will be applicable in this computation step and in all succeeding ones thus keeping the system from halting. Hence, we conclude* $N(\Pi') = Ps(L(G))$.

Instead of using the second channel we can also use a third cell instead for providing the trapping facility:

For the tissue P system

$$\Pi = \left(3, \{a\}, \lambda, a^2, \lambda, ch, R_{(2,0)}, R_{(2,1)}, R_{(2,3)}, R_{(3,0)}\right)$$

where $ch = \{(2,0), (2,1), (2,3), (3,0)\}$
with $R_{(2,0)} = \{a^{2i}/a^{2j+1} \mid X_i \to aX_j \in P\} \cup \{a^{2n}/\lambda\}$, $R_{(2,1)} = R_{(2,3)} = \{a/\lambda\}$, *and* $R_{(3,0)} = \{a/a\}$ *we obviously obtain* $N(\Pi) = N(\Pi') = Ps(L(G))$.

6.4.2 At Least Two Symbols and at Least Two Cells

As we are going to prove in this subsection, there seems to be a trade-off between the number of cells and the number of symbols: as our main result, we show that in the case of allowing two channels between a cell and the environment (we can restrict ourselves to only one channel between cells) computational completeness can be obtained with two cells and three symbols as well as with three cells and two symbols, respectively. We first show that when allowing only two symbols we need at most three cells for obtaining computational completeness, even for the case of only one channel between two cells as well as between a cell and the environment:

Theorem 6.4.4 $NRE = NO_2tP_3 = NO_2t'P_3$, *i.e.,*
$NRE = NO_ntP_m = NO_nt'P_m$ *for all* $n \geq 2$ *and* $m \geq 3$.

Proof. We only have to prove $NRE \subseteq NO_2tP_3$.

Let us consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; we now construct the tissue P system (of degree 3)

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\Pi = \left(3, \{a_1, p\}, \{a_1\}, w_1, \lambda, \lambda, ch, (R(i,j))_{(i,j)\in ch}\right),$$
$$ch = \{(1,0), (1,2), (2,0), (2,3), (3,0), (3,1)\}$$

which simulates the actions of M in such a way that Π halts if and only if M halts, thereby representing the final contents of register 1 of M by the corresponding multisets of symbols a_1 in the first cell (and no other symbols contained there). Throughout the computation, cell i of Π represents the contents of register i by the corresponding number of symbols a_1, whereas specific numbers of the symbols p represent the instructions to be simulated; moreover, p also has the function of a trap symbol, i.e., in case of the wrong choice for a rule to be applied we take in so many symbols p that we can never again rid of them and therefore get "trapped" in an infinite loop in cell 2.

An important part of the proof is to define a suitable encoding c for the instructions of the register machine: Without loss of generality we assume the labels of M to be positive integers such that the labels assigned to ADD and SUB instructions have the values $di + 1$ for $0 \leq i < t$, as well as $l_0 = 1$ and $l_h = d(t-1) + 1$, for some $t \geq 1$ and some constant $d > 1$ which allows us to have d consecutive codes for each instruction. As we shall see, in this proof it suffices to take $d = 7$.

We now define the encoding c on non-negative integers in such a way that $c : \mathbb{N} \to \mathbb{N}$ is a linear function that has to obey to the following additional conditions:

- For any i, j with $1 \leq i, j < dt$, $c(i) + c(j) > c(dt)$, i.e., the sum of the codes of two instruction labels has to be larger than the largest code $c(l_h)$ (observe that by assumption we have $l_h = d(t-1) + 1 < dt$ for $d > 1$) we will ever use for the given M.

- The distance g between any two codes $c(i)$ and $c(i+1)$ has to be large enough to allow one copy of the symbol p to be used for appearance checking as well as to allow specific numbers between 2 and $g - 1$ of copies of p to detect an incorrect application of rules. As we shall see in the construction of the rules below we use at most two copies of p in that way, hence, we take

  $$g = 3.$$

A function c fulfilling all the conditions stated above then, for example, is

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$c(x) = gx + gdt = 3x + 21t \text{ for } x \geq 0.$$

With $l_0 = 1$ we therefore obtain

$$c(l_0) = 21t + 3 \text{ and } w_1 = p^{c(l_0)} = p^{21t+3}.$$

Finally, we have to find a number f which is so large that after the introduction of f symbols we inevitably enter an infinite loop with the rule p^2/p^f; as we shall see below, the sum of all copies of the symbol p that may leave the "trap" cell 2 in any computation step is bounded by $2 + c(d(t-1)+1) < c(dt)$, hence, we may take

$$f = 2c(dt).$$

Equipped with this coding function c and the constants defined above we are now able to define the following sets of symport / antiport rules for simulating the actions of the given register machine M:

$$\begin{aligned}
R_{(1,0)} &= \left\{p^{c(l_1)}/p^{c(l_2)}a_1, p^{c(l_1)}/p^{c(l_3)}a_1 \mid l_1 : (A(1), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_1+1)}a_1, p^{c(l_1+2)}/p^{c(l_2)}, p^{c(l_1+2)}/p^{c(l_3)} \mid \right. \\
&\quad \left. l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_1+1)+1}, p^{c(l_1+2)}/p^{c(l_1+3)}, p^{c(l_1+4)}/p^{c(l_3)}, \right. \\
&\quad p^{c(l_1)}/p^{c(l_1+5)}, p^{c(l_1+6)}/p^{c(l_2)} \mid \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^{c(l_h)}/\lambda\right\}, \\
R_{(1,2)} &= \left\{p^{c(l_1+1)}a_1/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^{c(l_1+1)+1}/\lambda, p^{c(l_1+3)}/\lambda, p^{c(l_1+5)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^2/\lambda\right\}, \\
R_{(2,0)} &= \left\{pa_1/p^f, p^2/p^f\right\}, \\
R_{(2,3)} &= \left\{p^{c(l_1+1)}/\lambda \mid l_1 : (A(2), l_2, l_3) \in R\right\} \\
&\quad \left\{p^{c(l_1+1)}a_1/\lambda \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1+1)}/\lambda, p^{c(l_1+3)+1}/\lambda, p^{c(l_1+5)}a_1/\lambda \mid \right. \\
&\quad \left. l_1 : (S(2), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1+1)+1}/\lambda, p^{c(l_1+3)}/\lambda, p^{c(l_1+5)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\} \\
&\cup \left\{\lambda/p^2\right\},
\end{aligned}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned}
R_{(3,0)} &= \left\{p^{c(l_1+1)}/p^{c(l_1+2)} \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^{c(l_1+1)}/p^{c(l_1+2)}, p^{c(l_1+3)+1}/p^{c(l_1+4)},\right. \\
&\quad \left. p^{c(l_1+5)}a_1/p^{c(l_1+6)}, l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{pa_1/p^2\right\}, \\
R_{(3,1)} &= \left\{p^{c(l_1+2)}/\lambda \mid l_1 : (A(r), l_2, l_3) \in R, r \in \{2,3\}\right\} \\
&\cup \left\{p^{c(l_1+2)}/\lambda, p^{c(l_1+4)}/\lambda, p^{c(l_1+6)}/\lambda \mid\right. \\
&\quad \left. l_1 : (S(r), l_2, l_3) \in R, r \in \{2,3\}\right\}.
\end{aligned}$$

The correct work of the rules in Π can be described as follows:

1. Throughout the whole computation in Π, it is directed by the code $p^{c(l)}$ for some $l \leq l_h$; in order to guarantee the correct sequence of encoded rules, superfluous symbols p in case of a wrong choice guarantee an infinite loop with the symbols p by the "trap rule"

   $$p^2/p^f$$

   in $R_{(2,0)}$; the channel between cell 2 and the environment is only used to bring in f symbols p which guarantees that after the application of such a rule the number of symbols p will be increased in every further step, hence, the computation will never stop.

   As we shall elaborate in the following, finally cell 2 will be activated to work as a "trap" in the way described before whenever the wrong rule is chosen and superfluous copies of p remain to be used for "trapping" the whole system.

2. Each ADD instruction $l_1 : (A(1), l_2, l_3)$ of M is directly simulated by the rules

   $$p^{c(l_1)}/p^{c(l_2)}a_1 \text{ and}$$
   $$p^{c(l_1)}/p^{c(l_3)}a_1$$

   in $R_{(1,0)}$ in one step. The ADD instructions $l_1 : (A(r), l_2, l_3)$ of M, $r \in \{2,3\}$, are simulated in six steps in such a way that the new copy of symbol a_1 is transported to the corresponding cell r and, moreover, in cell 3 the code $c(l_1)$ is exchanged with the code $c(l_1+1)$ in order to guarantee that when returning to cell 1 a different code arrives which does not allow for misusing a symbol a_1 representing the contents of register 1 in cell 1 to start a new cycle with the original code. For example, the simulation of an ADD instruction $l_1 : (A(2), l_2, l_3)$ is accomplished by applying the following sequence of rules:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$p^{c(l_1)}/p^{c(l_1+1)}a_1$ from $R_{(1,0)}$,
$p^{c(l_1+1)}a_1/\lambda$ from $R_{(1,2)}$,
$p^{c(l_1+1)}/\lambda$ from $R_{(2,3)}$,
$p^{c(l_1+1)}/p^{c(l_1+2)}$ from $R_{(3,0)}$,
$p^{c(l_1+2)}/\lambda$ from $R_{(3,1)}$,
$p^{c(l_1+2)}/p^{c(l_2)}$ or $p^{c(l_1+2)}/p^{c(l_3)}$ from $R_{(1,0)}$.

If we do not choose one of the correct rules, then an infinite loop finally will be entered by the rule p^2/p^f:

- If in cell 1 a rule with the environment, i.e., from $R_{(1,0)}$, is used which is not correct, the remaining copies of p have to use the rule p^2/λ from $R_{(1,2)}$, as the coding function has been chosen in such a way that instead of the correct rule for a label l only rules for labels $l' < l$ could be chosen, whereas on the other hand, the number of symbols p is not large enough for allowing the remaining rest being interpreted as the code of another instruction label. Then in the next step cell 2 will be overflowed by the application of the rule p^2/p^f.
- If a rule from $R_{(1,2)}$ is used which is not correct, then the remaining copies of p in cell 1 will only allow p^2/λ from $R_{(1,2)}$ to be applied in the next step leading to the same consequences as described before.
- If in cell 2 a rule from $R_{(2,3)}$ is used which is not correct, then the remaining copies of p will enforce the application of the "trapping rule" p^2/p^f.
- If in cell 3 a rule from $R_{(3,0)}$ or $R_{(3,1)}$ is used which is not correct, then the remaining copies of p will enforce the application of the rule λ/p^2 from $R_{(2,3)}$, which then will cause unbounded growth of the number of symbols p in cell 2.

3. For simulating the decrementing step of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $p^{c(l_1+5)}$ to the corresponding cell r, where a correct continuation is only possible if this cell contains at least one symbol a_1. For example, decrementing register 2 is accomplished by applying the following sequence of six rules:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$p^{c(l_1)}/p^{c(l_1+5)}$ from $R_{(1,0)}$,
$p^{c(l_1+5)}/\lambda$ from $R_{(1,2)}$,
$p^{c(l_1+1)}a_1/\lambda$ from $R_{(2,3)}$,
$p^{c(l_1+5)}a_1/p^{c(l_1+6)}$ from $R_{(3,0)}$,
$p^{c(l_1+6)}/\lambda$ from $R_{(3,1)}$,
$p^{c(l_1+6)}/p^{c(l_2)}$ from $R_{(1,0)}$.

Again we notice that if at some moment we do not choose the correct rule, then finally we will enter an infinite loop overflowing cell 2 with copies of the symbol p.

4. For simulating the zero test, i.e., the case where the contents of register r is zero, of a SUB instruction $l_1 : (S(r), l_2, l_3)$ from R we send the code $p^{c(l_1+1)}$ together with one additional copy of the symbol p to cell r. In case our choice was wrong, i.e., register r is not empty, this additional symbol p for $r = 2$ will cause the application of the rule pa_1/p^f from $R_{(2,0)}$, which will immediately lead to an infinite computation in cell 2, or for $r = 3$ the application of the rule pa_1/p^2 from $R\,(3, 0)$ will enforce the application of the rule λ/p^2 from $R_{(2,3)}$ in the next step, which then will cause unbounded growth of the number of symbols p in cell 2. If our choice was correct, i.e., register r is empty, then in cell 3, the code $p^{c(l_1+1)}$ is exchanged with the code $p^{c(l_1+2)}$, which is exchanged with $p^{c(l_1+3)}$ in cell 1. This code $p^{c(l_1+3)}$ then captures the additional symbol p left back in cell r, and in cell 3 this additional symbol p goes out together with code $p^{c(l_1+3)}$, instead, code $p^{c(l_1+4)}$ continues and in cell 1 allows for replacing it with $p^{c(l_2)}$. For example, for testing register 3 for zero we take the following rules:

$p^{c(l_1)}/p^{c(l_1+1)+1}$ from $R_{(1,0)}$,
$p^{c(l_1+1)+1}/\lambda$ from $R_{(1,2)}$,
$p^{c(l_1+1)+1}/\lambda$ from $R_{(2,3)}$,
$p^{c(l_1+1)}/p^{c(l_1+2)}$ from $R_{(3,0)}$,
$p^{c(l_1+2)}/\lambda$ from $R_{(3,1)}$,
$p^{c(l_1+2)}/p^{c(l_1+3)}$ from $R_{(1,0)}$,
$p^{c(l_1+3)}/\lambda$ from $R_{(1,2)}$,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$p^{c(l_1+3)}/\lambda$ from $R_{(2,3)}$,
$p^{c(l_1+3)+1}/p^{c(l_1+4)}$ from $R_{(3,0)}$,
$p^{c(l_1+4)}/\lambda$ from $R_{(3,1)}$,
$p^{c(l_1+4)}/p^{c(l_3)}$ from $R_{(1,0)}$.

Once again we notice that if at some moment we do not choose the correct rule, then this will cause a non-halting computation by overflowing cell 2 with copies of symbol p.

5. Finally, for the halt label l_h we take the rule

   $p^{c(l_h)}/\lambda$,

   hence, the work of Π will stop exactly when the work of M stops (provided the system has not become overflowed by symbols p due to a wrong non-deterministic choice during the computation).

From the explanations given above we conclude that Π halts if and only if M halts, and moreover, the final configuration of Π represents the final contents of the registers in M. These observations conclude the proof. □

When the number of objects is increased to three, we need only two cells provided we have two channels between a cell and the environment:

Theorem 6.4.5 $NRE = NO_3t'P_2$, i.e.,
$NRE = NO_nt'P_m$ for all $n \geq 3$ and $m \geq 2$.

Proof. We only prove $NRE \subseteq NO_3t'P_2$.

As in the previous proof, we consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; we now construct the tissue P system (of degree 2)

$$\Pi = \left(2, \{a_1, a_2, p\}, \{a_1\}, w_1, \lambda, ch, \left(R_{(i,j)}\right)_{(i,j)\in ch}\right),$$
$$ch = \{(0,1), (0,2), (1,0), (1,2), (2,0)\},$$

which simulates the actions of M in such a way that throughout the computation, specific numbers of the symbols p represent the instructions to be simulated, the number of symbols a_1 in cell 1 of Π represents the contents of register 1 by the corresponding number of symbols a_1 and the new symbol

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

a_2 represents the contents of registers 2 and 3 by the corresponding number of symbols a_2 in cells 1 and 2, respectively.

We shall use a similar encoding c as in the previous proof; as only five consecutive labels for each instruction of M are needed, it suffices to take $d = 5$; moreover, we only need one copy of p for the zero tests and two copies for detecting wrong applications of rules, therefore we take $g = 3$. Thus, we now use the function c with

$$c(x) = gx + gdt = 3x + 15t \text{ for } x \geq 0.$$

With $l_0 = 1$ we therefore obtain

$$c(l_0) = 15t + 3 \text{ and } w_1 = p^{c(l_0)} = p^{15t+3}.$$

If something goes wrong, the additional channels $R_{(0,1)}$ and $R_{(0,2)}$ allow for "trapping" the system by introducing f copies of p; for f we now take $3c(dt)$.

For simulating the actions of the given register machine M, we define the following sets of symport / antiport rules:

$$\begin{aligned}
R_{(0,1)} &= \left\{p^f/p^2, p^f/pa_2\right\}, \\
R_{(0,2)} &= \left\{p^f/p^2, p^f/pa_2\right\}, \\
R_{(1,0)} &= \left\{p^{c(l_1)}/p^{c(l_2)}a_1, p^{c(l_1)}/p^{c(l_3)}a_1 \mid l_1 : (A(1), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_2)}a_2, p^{c(l_1)}/p^{c(l_3)}a_2 \mid l_1 : (A(2), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_1+1)}a_2, p^{c(l_1+2)}/p^{c(l_2)}, p^{c(l_1+2)}/p^{c(l_3)} \mid \right. \\
&\quad \left. l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}a_2/p^{c(l_2)}, p^{c(l_1)}/p^{c(l_1+1)+1}, p^{c(l_1+1)}/p^{c(l_1+2)}, \right. \\
&\quad \left. p^{c(l_1+2)+1}/p^{c(l_3)} \mid l_1 : (S(2), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1+1)}/p^{c(l_2)}, p^{c(l_1+4)}/p^{c(l_3)} \mid \right. \\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_h)}/\lambda\right\}, \\
R_{(1,2)} &= \left\{p^{c(l_1+1)}a_2/\lambda, \lambda/p^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}/\lambda, \lambda/p^{c(l_1+1)}/\lambda, p^{c(l_1+4)}/\lambda \mid \right. \\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\}, \\
R_{(2,0)} &= \left\{p^{c(l_1+1)}/p^{c(l_1+2)} \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}a_2/p^{c(l_1+1)}, p^{c(l_1)}/p^{c(l_1+2)+1}, p^{c(l_1+2)}/p^{c(l_1+3)}, \right. \\
&\quad \left. p^{c(l_1+3)+1}/p^{c(l_1+4)} \mid l_1 : (S(3), l_2, l_3) \in R\right\}.
\end{aligned}$$

The simulation of the instructions of M by Π now works as follows:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

1. Each ADD instruction $l_1 : (A(i), l_2, l_3)$ of M, $1 \leq i \leq 2$, is directly simulated by the rules

   $p^{c(l_1)}/p^{c(l_2)}a_i$ or $p^{c(l_1)}/p^{c(l_3)}a_i$

   from $R_{(1,0)}$ in one step. As the first register is never decremented, these are the only rules involving the symbol a_1.

2. Each ADD instruction $l_1 : (A(3), l_2, l_3)$ of M is simulated by the following sequence of rules:

   $p^{c(l_1)}/p^{c(l_1+1)}a_2$ from $R_{(1,0)}$,
   $p^{c(l_1+1)}a_2/\lambda$ from $R_{(1,2)}$,
   $p^{c(l_1+1)}/p^{c(l_1+2)}$ from $R_{(2,0)}$,
   $\lambda/p^{c(l_1+2)}$ from $R_{(1,2)}$,
   $p^{c(l_1+2)}/p^{c(l_2)}$ or $p^{c(l_1+2)}/p^{c(l_3)}$from $R_{(1,0)}$.

3. For simulating the decrementing step of a SUB instruction $l_1 : (S(2), l_2, l_3)$ from R we simply use the rule $p^{c(l_1)}a_2/p^{c(l_2)}$ from $R_{(1,0)}$.

4. For simulating the decrementing step of a SUB instruction $l_1 : (S(3), l_2, l_3)$ from R we have to use the following sequence of rules:

   $p^{c(l_1)}/\lambda$ from $R_{(1,2)}$,
   $p^{c(l_1)}a_2/p^{c(l_1+1)}$ from $R_{(2,0)}$,
   $\lambda/p^{c(l_1+1)}$ from $R_{(1,2)}$,
   $p^{c(l_1+1)}/p^{c(l_2)}$ from $R_{(1,0)}$.

5. For simulating the zero test, i.e., the case where the contents of register 2 is zero, of a SUB instruction $l_1 : (S(2), l_2, l_3)$ from R we use the rules

   $p^{c(l_1)}/p^{c(l_1+1)+1}$,
   $p^{c(l_1+1)}/p^{c(l_1+2)}$, and
   $p^{c(l_1+2)+1}/p^{c(l_3)}$ from $R_{(1,0)}$.

   The first rule introduces an additional copy of p which in the succeeding step can be used for appearance checking; in the failure case, i.e., if at least one copy of a_2 is present in cell 1, then the rule p^f/pa_2 from $R_{(0,1)}$ has to be used thus "trapping" the whole system. Otherwise, this additional copy of p is eliminated in the third step by the rule $p^{c(l_1+2)+1}/p^{c(l_3)}$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

6. For simulating the zero test for register 3 in a SUB instruction $l_1 : (S(3), l_2, l_3)$ from R we have to use the following sequence of rules:

   $p^{c(l_1)}/\lambda$ from $R_{(1,2)}$,
   $p^{c(l_1)}/p^{c(l_1+2)+1}$ from $R_{(2,0)}$,
   $p^{c(l_1+2)}/p^{c(l_1+3)}$ from $R_{(2,0)}$,
   $p^{c(l_1+3)+1}/p^{c(l_1+4)}$ from $R_{(2,0)}$,
   $\lambda/p^{c(l_1+4)}$ from $R_{(1,2)}$,
   $p^{c(l_1+4)}/p^{c(l_3)}$ from $R_{(1,0)}$.

   In the failure case, i.e., if at least one copy of a_2 is present in cell 2, then the rule p^f/pa_2 from $R_{(0,2)}$ has to be used thus "trapping" the whole system.

7. Finally, for the halt label l_h we take the rule $p^{c(l_h)}/\lambda$ from $R_{(1,0)}$; hence, the work of Π will stop exactly when the work of M stops (provided none of the trap rules from $R_{(0,1)}$ has been applied due to a wrong non-deterministic choice during the computation).

As one can easily see, Π halts if and only if M halts, and moreover, in the final configuration of Π cell 1 represents the final contents of register 1 in M. If at some moment we do not use the correct rule, then an infinite loop will be entered by applying one of the rules from the sets $R_{(0,i)}$, $i \in \{1, 2\}$. These observations conclude the proof. □

For the case of only one channel between a cell and the environment, it remains to investigate how many symbols we need for obtaining computational completeness with only two cells; as the following theorem shows, four symbols are already sufficient:

Theorem 6.4.6 $NRE = NO_4tP_2$, i.e.,
$NRE = NO_ntP_m$ for all $n \geq 4$ and $m \geq 2$.

Proof. We only have to prove $NRE \subseteq NO_4tP_2$. Again we start from a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$. The main ideas for the construction of the tissue P system with two cells and four symbols

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\Pi = \left(2, \{a_1, a_2, p, q\}, w_1, q, ch, \left(R_{(i,j)}\right)_{(i,j)\in ch}\right),$$
$$ch = \{(1,0), (1,2), (2,0)\},$$

generating L are the following ones:

The contents of register 1 is stored as the number of symbols a_1 in the first cell, whereas the numbers of symbols a_2 represent the contents of register 2 and register 3, respectively. The fourth symbol q is only used as a trap symbol. We start with one symbol q already being present in cell 2. Checking for the appearance of a symbol a_2 in cell 1 or cell 2 involves one symbol p which in the presence of a symbol a_2 allows for transporting this symbol q from cell 2 to cell 1, where it will cause an infinite computation by the rule $q/q \in R_{(1,0)}$.

The instructions again are encoded by a suitable function $c : \mathbb{N} \to \mathbb{N}$; as we use one symbol p for appearance checking, we need (at least) two symbols p for detecting a wrong application of rules (which will cause the introduction of a trap symbol q in one of the two cells, thus guaranteeing the derivation never to halt), hence, the distance between two codes $c(i)$ and $c(i+1)$ has to be at least 3. As we shall see from the construction given below, we have (at most) five consecutive codes for each of the ADD and SUB instructions, which, without loss of generality, we assume to have the labels $5i+1$ for $0 \leq i < t-1$, as well as $l_0 = 1$ and $l_h = 5(t-1)+1$, for some $t > 1$. Moreover, as in the proofs of the preceding theorems, for any i, j with $1 \leq i, j < 5t$, we demand $c(i) + c(j) > c(5t)$. Hence, in this proof we take

$$c(x) = 3x + 15t \text{ for } x \geq 0.$$

With $l_0 = 1$ we therefore obtain $c(l_0) = 15t + 3$ and $w_1 = b^{c(l_0)} = b^{15t+3}$.

The rules in the channels between the two cells and the environment as well as in the channel from cell 1 to cell 2 now are defined as follows:

$$\begin{aligned}
R_{(1,0)} &= \left\{p^{c(l_h)}/\lambda\right\} \cup \{p^2/q, q/q\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_2)}a_1, p^{c(l_1)}/p^{c(l_3)}a_1 \mid l_1 : (A(1), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_2)}a_2, p^{c(l_1)}/p^{c(l_3)}a_2 \mid l_1 : (A(2), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1+1)}/p^{c(l_2)}, p^{c(l_1+1)}/p^{c(l_3)} \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}a_2/p^{c(l_2)}, p^{c(l_1)}/p^{c(l_1+1)+1}, p^{c(l_1+1)}/p^{c(l_1+2)},\right. \\
&\quad \left. p^{c(l_1+2)+1}/p^{c(l_3)} \mid l_1 : (S(2), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1+1)}/p^{c(l_2)}, p^{c(l_1+4)}/p^{c(l_3)} \mid l_1 : (S(3), l_2, l_3) \in R\right\},
\end{aligned}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned}
R_{(1,2)} &= \{\lambda/p^2q, \lambda/pqa_2, p^2/q, pa_2/q\} \\
&\cup \left\{p^{c(l_1)}/\lambda, \lambda/p^{c(l_1+1)} \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\quad \left\{p^{c(l_1)}/\lambda, \lambda/p^{c(l_1+1)}, \lambda/p^{c(l_1+4)} \mid \right. \\
&\quad \left. l_1 : (S(3), l_2, l_3) \in R\right\} \\
R_{(2,0)} &= \{p^2/q, q^2/q^2\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_1+1)}a_2 \mid l_1 : (A(3), l_2, l_3) \in R\right\} \\
&\cup \left\{p^{c(l_1)}a_2/p^{c(l_1+1)}, p^{c(l_1)}/p^{c(l_1+2)+1}, p^{c(l_1+2)}/p^{c(l_1+3)}, \right. \\
&\quad \left. p^{c(l_1+3)+1}/p^{c(l_1+4)} \mid l_1 : (S(3), l_2, l_3) \in R\right\}.
\end{aligned}$$

We now describe how the rules in Π correctly simulate the actions of M:

1. Throughout the whole computation in Π, the application of rules is directed by the code $p^{c(l)}$ for some $l \leq l_h$; in order to guarantee the correct sequence of encoded rules, superfluous symbols p in case of a wrong choice guarantee an infinite loop with the trap symbol q by the "trap rules" q/q in $R_{(1,0)}$ and q^2/q^2 in $R_{(2,0)}$. Whereas the rules p^2/q in $R_{(1,0)}$ and $R_{(2,0)}$ can take an additional trap symbol from the environment, cell 1 has to take the trap symbol q from cell 2 if either superfluous symbols p remain when cell 1 uses a wrong rule in $R_{(1,0)}$ or if a symbol a_2 is present in cell 1 when register 2 is tested for zero. In a similar way, cell 2 sends its initial copy of q to cell 1 if it uses a wrong rule from $R_{(2,0)}$ or if a symbol a_2 is present in cell 2 when register 3 is tested for zero.

2. Each ADD instruction $l_1 : (A(1), l_2, l_3)$ of M is directly simulated by the rules

   $$p^{c(l_1)}/p^{c(l_2)}a_1 \text{ or } p^{c(l_1)}/p^{c(l_3)}a_1$$

   in $R_{(1,0)}$ in one step. As the first register is never decremented, these are the only rules involving the symbol a_1.

3. Each ADD instruction $l_1 : (A(2), l_2, l_3)$ of M is directly simulated by the rules

   $$p^{c(l_1)}/p^{c(l_2)}a_2 \text{ or } p^{c(l_1)}/p^{c(l_3)}a_2$$

   in $R_{(1,0)}$ in one step.

4. Each ADD instruction $l_1 : (A(3), l_2, l_3)$ of M is simulated by the sequence of rules

   $$p^{c(l_1)}/\lambda \text{ in } R_{(1,2)},$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$p^{c(l_1)}/p^{c(l_1+1)}a_2$ in $R_{(2,0)}$,
$\lambda/p^{c(l_1+1)}$ in $R_{(1,2)}$, and
$p^{c(l_1+1)}/p^{c(l_2)}$ or $p^{c(l_1+1)}/p^{c(l_3)}$ in $R_{(1,0)}$.

5. For simulating the decrementing step of a SUB instruction $l_1 : (S(2), l_2, l_3)$ from R we simply use the rule $p^{c(l_1)}a_2/p^{c(l_2)}$ in $R_{(1,0)}$.

6. For simulating the zero test, i.e., the case where the contents of register 2 is zero, of a SUB instruction $l_1 : (S(2), l_2, l_3)$ from R we use the rule $p^{c(l_1)}/p^{c(l_1+1)+1}$ in $R_{(1,0)}$. In the next step, we use the rule $p^{c(l_1+1)}/p^{c(l_1+2)}$ in $R_{(1,0)}$, whereas the additional symbol p is using for the zero test: if a symbol a_2 is present in cell 1, then this one additional copy of p together with one copy of a_2 exchanges with the trap symbol q by the rule pa_2/q in $R_{(1,2)}$. Only if no symbol a_2 is present, the additional copy of p will be erased by using the rule $p^{c(l_1+2)+1}/p^{c(l_3)}$ in $R_{(1,0)}$, which ends the successful simulation of the zero test.

7. For simulating the SUB instruction $l_1 : (S(3), l_2, l_3)$ from R we send the code $p^{c(l_1)}$ to cell 2. There we either simulate a decrementing step by using the rule $p^{c(l_1)}a_2/p^{c(l_1+1)}$ in $R_{(2,0)}$ (then followed by the rule $\lambda/p^{c(l_1+1)}$ in $R_{(1,2)}$ and the rule $p^{c(l_1+1)}/p^{c(l_2)}$ in $R_{(1,0)}$) or the zero test by using the sequence of rules $p^{c(l_1)}/p^{c(l_1+2)+1}$, $p^{c(l_1+2)}/p^{c(l_1+3)}$, and $p^{c(l_1+3)+1}/p^{c(l_1+4)}$ in $R_{(2,0)}$. The additional copy of p after the application of the first rule of this sequence again is used for checking for the presence of a symbol a_2, now in cell 2. Only if no symbol a_2 is present, the simulation of the zero test successfully continues with the application of the rule $\lambda/p^{c(l_1+4)}$ in $R_{(1,2)}$ and ends with the application of the rule $p^{c(l_1+4)}/p^{c(l_3)}$ in $R_{(1,0)}$. On the other hand, the zero test fails if the additional copy of p together with a symbol a_2 can take the trap symbol q to cell 1 by the rule λ/pqa_2 in $R_{(1,2)}$.

8. Finally, for the halt label l_h we take the rule $p^{c(l_h)}/\lambda$ from $R_{(1,0)}$; hence, the work of Π will stop exactly when the work of M stops and the computation in M has been correctly simulated by Π. On the other hand, if during the simulation of the computation in M by Π some error has occurred, this inevitably has led to the introduction of a trap symbol q in cell 1 or a second copy of the trap symbol q in cell 2, which in both cases prevents Π from halting. Thus, we conclude $N(\Pi) = L$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

□

6.4.3 One Cell

In this section we investigate the remaining variant of using only one cell, in which case it turns out that the definition of the tissue P system is essential, i.e., computational completeness can only be obtained with two channels between the cell and the environment, whereas we can only generate regular sets when using only one channel between the cell and the environment.

The proof of the following completeness result can be obtained following the construction given in [13] for P systems:

Theorem 6.4.7 *$NRE = NO_n t' P_1$ for all $n \geq 5$.*

Proof. We only have to prove $NRE \subseteq NO_5 t' P_1$. Let us consider a register machine $M = (3, R, l_0, l_h)$ with three registers generating $L \in NRE$; the main ideas for the construction of the tissue P system with one cell (and two channels between the cell and the environment) and five symbols

$$\Pi = \left(1, \{a_1, a_2, a_3, p, q\}, w_1, \{(0,1), (1,0)\}, R_{(0,1)}, R_{(1,0)}\right)$$

generating L are the following ones:

The symbols a_1, a_2, and a_3 represent the three registers; the symbols p and q are needed for encoding the instructions of M; q also has the function of a trap symbol, i.e., in case of the wrong choice for a rule to be applied we take in so many symbols q that we can never again rid of them and therefore get "trapped" in an infinite loop.

The (labels of the) instructions of the register machine M will be encoded as suitable numbers of the symbol p. Moreover, we will use a combination of small numbers of symbols p and q to be able to selectively check for the appearance of a_2 and a_3, i.e., for testing register 2/ register 3 for zero we take pq^2 and p^2q, respectively.

The instructions again are encoded by a suitable function $c : \mathbb{N} \to \mathbb{N}$; as we use one or two symbols p for appearance checking, we need (at least) three symbols p for detecting a wrong application of rules (which will cause the introduction of trap symbols q in the cells, thus guaranteeing the derivation never to halt), hence, the distance between two codes $c(i)$ and $c(i+1)$ has to

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

be at least 5. As we shall see from the construction given below, we have (at most) three consecutive codes for each of the ADD and SUB instructions, which, without loss of generality, we assume to have the labels $3i+1$ for $0 \leq i < t-1$, as well as $l_0 = 1$ and $l_h = 3(t-1)+1$, for some $t > 1$. Moreover, as in the proofs of the preceding theorems, for any i, j with $1 \leq i, j < 3t$, we demand $c(i) + c(j) > c(3t)$. Hence, in this proof we take

$$c(x) = 5x + 15t \text{ for } x \geq 0.$$

With $l_0 = 1$ we therefore obtain $c(l_0) = 15t + 5$ and $w_1 = p^{c(l_0)} = p^{15t+5}$.

Finally, we have to find a number f which is so large that after the introduction of f symbols q we inevitably enter an infinite loop with the rule q^{2f}/q^f in $R_{(0,1)}$; as at most two symbols q are used for encoding the zero tests, we can take $f = 3$.

The rules in the two channels between the cell and the environment now are defined as follows:

$$\begin{aligned}
R_{(1,0)} &= \left\{p^{c(l_h)}/\lambda\right\} \\
&\cup \left\{p^{c(l_1)}/p^{c(l_2)}a_i, p^{c(l_1)}/p^{c(l_3)}a_i \mid \right. \\
&\quad \left. l_1 : (A(i), l_2, l_3) \in R, 1 \leq i \leq 3\right\} \\
&\cup \left\{p^{c(l_1)}a_k/p^{c(l_2)}, p^{c(l_1)}/p^{c(l_1+1)}p^{k-1}q^{3-k}, \right. \\
&\quad p^{c(l_1+1)}/p^{c(l_1+2)}, p^{c(l_1+2)}p^{k-1}q^{3-k}/p^{c(l_3)} \mid \\
&\quad \left. l_1 : (S(k), l_2, l_3) \in R, 2 \leq k \leq 3\right\}, \\
R_{(0,1)} &= \left\{q^6/q^3, q^6/p^3, q^6/pq^2a_2, q^6/p^2qa_3\right\}.
\end{aligned}$$

We now describe how the rules in Π correctly simulate the actions of M:

1. Throughout the whole computation in Π, the application of rules is directed by the code $p^{c(l)}$ for some $l \leq l_h$, and the corresponding rules in $R_{(1,0)}$ should guarantee the correct sequence of encoded rules. The rules in $R_{(0,1)}$ are only applied if something goes wrong, i.e., either superfluous symbols p in case of a wrong choice of the rule from $R_{(1,0)}$ cause the application of the rule q^6/p^3 from $R_{(0,1)}$ or the failure of the zero test causes the application of the rule q^6/pq^2a_2 or q^6/p^2qa_3 in case a_2 or a_3 is present. The rule q^6/q^3 in $R_{(0,1)}$ guarantees that as soon as six trap symbols q have been introduced by some rule in $R_{(0,1)}$, then we can never get rid of them again, their number steadily increases, because at most two of them can leave through channel $(1,0)$ using a rule $p^{c(l_1+2)}p^{k-1}q^{3-k}/p^{c(l_3)}$.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

2. Each ADD instruction $l_1 : (A(i), l_2, l_3)$ of M is directly simulated by the rules

   $$p^{c(l_1)}/p^{c(l_2)}a_i \text{ or } p^{c(l_1)}/p^{c(l_3)}a_i$$

   in $R_{(1,0)}$ in one step. As the first register is never decremented, these are the only rules involving the symbol a_1.

3. For simulating the decrementing step of a SUB instruction $l_1 : (S(k), l_2, l_3)$ from R we simply use the rule $p^{c(l_1)}a_k/p^{c(l_2)}$ in $R_{(1,0)}$, $2 \leq k \leq 3$.

4. For simulating the zero test, i.e., the case where the contents of register k is zero, of a SUB instruction $l_1 : (S(k), l_2, l_3)$ from R, $2 \leq k \leq 3$, we first use the rule $p^{c(l_1)}/p^{c(l_1+1)}p^{k-1}q^{3-k}$ in $R_{(1,0)}$. In the next step, we use the rule $p^{c(l_1+1)}/p^{c(l_1+2)}$ in $R_{(1,0)}$, whereas the additional symbols $p^{k-1}q^{3-k}$ are used for the zero test: if a symbol a_k is present in the cell, then $p^{k-1}q^{3-k}$ together with one copy of a_k introduces the deadly number of six trap symbols q by the rule $q^6/p^{k-1}q^{3-k}a_k$. Only if no symbol a_k is present, $p^{k-1}q^{3-k}$ will be erased by using the rule $p^{c(l_1+2)}p^{k-1}q^{3-k}/p^{c(l_3)}$ in $R_{(1,0)}$, which ends the successful simulation of the zero test.

5. Finally, for the halt label l_h we take the rule $p^{c(l_h)}/\lambda$ from $R_{(1,0)}$; hence, the work of Π will stop exactly when the work of M stops (provided none of the trap rules from $R_{(0,1)}$ has been applied due to a wrong non-deterministic choice during the computation).

From the explanations given above we conclude that Π halts if and only if M halts, and moreover, the final configuration of Π represents the final contents of the registers in M. These observations conclude the proof. □

Remark 6.4.1 *We should like to mention that the preceding theorem immediately implies $NRE = NO_5tP_2$, yet Theorem 6.4.6 has already provided us with the better result $NRE = NO_4tP_2$, i.e., we need at most four symbols for obtaining computational completeness with two cells.*

As it has already been proved in Theorem 6.4.3,

$$NFIN = NO_1tP_1 = NO_1t'P_1.$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

The following result shows that with only one cell and one channel between the single cell and the environment exactly the regular sets can be generated:

Theorem 6.4.8 *$NREG = NO_ntP_1$ for all $n \geq 2$.*

Proof. We first prove $NREG \subseteq NO_2tP_1$ thereby using the fact that for any regular set M of non-negative integers there exist finite sets of numbers M_0, M_1 and a number k such that $M = M_0 \cup \{i + jk \mid i \in M_1, j \in \mathbb{N}\}$ (this follows, e.g., from the shape of the minimal finite automaton accepting the unary language with length set M).

We now construct a tissue P system $\Pi = (1, \{a, p\}, pp, \{(1, 0)\}, R_{(1,0)})$ where $R_{(1,0)} = \{pp/a^i \mid i \in M_0\} \cup \{pp/pa, pa/pa^{k+1}\} \cup \{a/a^i \mid i \in M_1\}$, which generates M as the number of symbols a in the cell in halting computations.

Initially, there are no objects a in the cell, so the system "chooses" between generating an element of M_0 in one step or exchanging pp by pa. In the latter case, there is only one copy of p in the system. After an arbitrary number j of applications of the rule pa/pa^{k+1} a rule exchanging pa by a^i for some $i \in M_1$ is eventually applied, generating $jk + i$ symbols a. Hence, $N(\Pi) = M_0 \cup \{i + jk \mid i \in M_1, j \in \mathbb{N}\} = M$.

We now show that even with more than two symbols we cannot generate more than regular sets with one-cell tissue P systems using only one channel between the cell and the environment (in contrast to the case where we allow two channels, see Theorem 6.4.7). As is well known from [109], $NREG = PsMAT^{\lambda}(1)$ (i.e., $NREG$ equals the family of sets of natural numbers that can be generated by matrix grammars without appearance checking over a one-letter alphabet). Hence, it remains to show that $NO_ntP_1 \subseteq PsMAT^{\lambda}(1)$ (which can be done in a quite similar way as in the proof of Lemma 2 in [14]):

Consider the tissue P system

$$\Pi = \left(1, \{a_1, ..., a_n\}, w_1, \{(1, 0)\}, R_{(1,0)}\right).$$

A set M representing the reachable configurations, i.e., the possible multisets contained in the single cell during any computation in the tissue P system can easily be generated by a matrix grammar without appearance checking:

Let $h : \{a_1, ..., a_n\} \to \{\bar{a}_1, ..., \bar{a}_n\}$ be the renaming morphism defined by $h(a_i) = \bar{a}_i$ for $1 \leq i \leq n$. Moreover, for any production p, in a matrix let

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$(p,)^k$ denote the sequence of k times repeating the production p; for $k = 0$ this just stands for the empty string (not contributing to the matrix). Then a matrix grammar generating the reachable configurations of Π (a reachable configuration is obtained as some string representing the multiset of objects in the cell) can be defined as follows:

$$\begin{aligned} G &= (N, T, S, M), \\ N &= \{S, X, Y\} \cup \{\bar{a}_1, \ldots, \bar{a}_n\}, \\ T &= \{a_1, \ldots, a_n\}, \\ M &= \{[S \to Xh(w_1)], [X \to Y], [Y \to \lambda]\} \\ &\cup \left\{ \left[(\bar{a}_1 \to \lambda,)^{k_1} \ldots (\bar{a}_n \to \lambda,)^{k_n} X \to X {\bar{a}_1}^{m_1} \ldots {\bar{a}_n}^{m_n} \right] \mid \right. \\ &\qquad \left. {a_1}^{k_1} \ldots {a_n}^{k_n} / {a_1}^{m_1} \ldots {a_n}^{m_n} \in R_{(1,0)} \right\} \\ &\cup \{[Y \to Y, \bar{a}_i \to a_i] \mid 1 \leq i \leq n\}. \end{aligned}$$

Moreover, the set K of strings representing the multisets to which a rule from $R_{(1,0)}$ can be applied is regular (e.g., think of a non-deterministic finite automaton guessing the rule $a_1^{k_1} \ldots a_n^{k_n} / a_1^{m_1} \ldots a_n^{m_n} \in R_{(1,0)}$ and then checking the number of symbols a_i in the underlying string to be at least k_i for all i, $1 \leq i \leq n$; we leave the details of this construction to the reader). Hence, the complement K^c of K is regular, too. Then the Parikh set of the projection of $M \cap K^c$ onto $\{a_1\}^*$ is exactly $N(\Pi)$; due to the closure properties of the family of languages generated by matrix grammars without appearance checking, $N(\Pi)$ therefore is in $PsMAT^{\lambda}(1)$, i.e., $N(\Pi) \in NREG$. □

6.4.4 Summary and Open Questions

In sum, for tissue P systems with only one channel between two cells and between a cell and the environment we could show the results listed in Table 6.2 and for tissue P systems with two channels between a cell and the environment we could show the results listed in Table 6.3. In both tables, $NFIN$, $NREG$, and NRE indicate the equality with the corresponding family NO_mtP_n and $NO_mt'P_n$, respectively; A indicates that the corresponding family includes at least $NREG$, and B indicates that the corresponding family strictly includes $NFIN$.

The main open question concerns a characterization of the sets of natural numbers in NO_2tP_2 and NO_2tP_3. Further, it would be interesting to find the minimal number l such that NO_1tP_l contains all recursively enumerable sets of natural numbers, whereas the families NO_1tP_j with $j < l$ do not fulfill this condition. Finally, it remains to find characterizations of the sets of

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Table 6.2: Families NO_ntP_m

symbols							
4	$NREG$	NRE	NRE	NRE	NRE	NRE	NRE
3	$NREG$	NRE	NRE	NRE	NRE	NRE	NRE
2	$NREG$	A	A	NRE	NRE	NRE	NRE
1	$NFIN$	B	A	A	A	A	NRE
	1	2	3	4	5	6	7 cells

natural numbers in those families NO_1tP_j that do not contain all recursively enumerable sets of natural numbers.

Table 6.3: Families $NO_nt'P_m$

symbols						
5	NRE	NRE	NRE	NRE	NRE	NRE
4	A	NRE	NRE	NRE	NRE	NRE
3	A	NRE	NRE	NRE	NRE	NRE
2	A	A	NRE	NRE	NRE	NRE
1	$NFIN$	A	A	A	A	NRE
	1	2	3	4	5	6 cells

The most interesting open problems for the families $NO_nt'P_m$ are to find the minimal number k as well as the minimal number l such that $NO_kt'P_1$ and $NO_1t'P_l$, respectively, contain all recursively enumerable sets of natural numbers, whereas the families $NO_it'P_1$ and $NO_1t'P_j$ with $i < k$ and $j < l$, respectively, do not fulfill this condition. Moreover, it remains to find characterizations of the sets of natural numbers in those families $NO_nt'P_m$ that do not contain all recursively enumerable sets of natural numbers.

Some of the proofs elaborated in this paper (e.g., the proof of Theorem 6.4.7) can easily be modified in order to show that the corresponding tissue P systems can also accept any set from NRE by just enlarging the system by rules for simulating SUB instructions on the first register. A thorough investigation of the descriptional complexity of tissue P systems for accepting NRE or generating / accepting even sets of vectors of natural numbers or for computing functions $\mathbb{N}^m \to \mathbb{N}^n$ as partially done for P systems in [16] remains for future research.

Related open problems concern the families NO_mP_n of sets of natural

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

numbers generated by P systems with symport / antiport rules as well as n symbols and m membranes. The first result proving computational completeness for P systems with three symbols and four membranes was obtained in [165] and continued in [13], where P systems with five symbols in only one membrane were shown to be computationally complete. A thorough investigation of the families NO_mP_n can be found in [16]. The main open problem in the case of P systems is the question whether one symbol is sufficient to obtain computational completeness as was shown for the case of tissue P systems in [94].

6.5 Concluding Remarks

The complete descriptional complexity results are summarized in Table 6.1 for P systems and in Tables 6.2, 6.3 for tissue P systems.

We now finish our overview with repeating (some of) the best known results of computational completeness. Both completeness results for one-symbol systems (shown in boldface) are obtained in [94], while the other completeness proofs, obtained in [13], [16], [17] and [15], were presented in this chapter.

P systems

$$NRE = NO_nP_m(sym_*, anti_*) \quad \text{for } (n,m) \in \{(5,1),(4,2),(3,3),(2,4)\}.$$

Tissue P systems

$$NRE = NO_ntP_m(sym_*, anti_*) \quad \text{for } (n,m) \in \{(4,2),(2,3),(\mathbf{1},\mathbf{7})\}.$$
$$NRE = NO_nt'P_m(sym_*, anti_*) \quad \text{for } (n,m) \in \{(5,1),(3,2),(2,3),(\mathbf{1},\mathbf{6})\}.$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 7

Applications

The results that will be presented in this chapter are based on articles [32], [33], [9] and [6].

7.1 Sorting

7.1.1 Introduction

The main topic of this chapter is application of P systems for sorting problems, i.e., constructing sorting algorithms for various variants of P systems. Traditional studies of sorting assume a constant time for comparing two numbers and compute the time complexity with respect to the number of components of a vector to be sorted. Here we assume the number of components to be a fixed number k, and study various algorithms with distinct generative features based on different models of P systems, and their time complexity with respect to the maximal number, or to their sum. Massively parallel computations that can be realized within the framework of P systems offer a premise to major improvements of the classical integer sorting problems. Despite this important characteristic we will see that, depending on the model used, the massive parallelism feature cannot be always used, and so, some results will have the complexity "comparable" with the classical integer sorting algorithms. Still, computing an (ordered) word from an (unordered) multiset can be a goal not only for computer science but also, at least, for bio-synthesis (separating mixed objects according to some characteristics). Here, we will pass from ranking algorithms that, starting with

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

numbers represented as multisets, produce symbols in an order, to effective sorting algorithms.

P systems with symbol objects turn out to be a convenient framework to describe computations based on molecular interaction. Many different variants of such P systems turn out to be computationally universal and so the reason for studying algorithms that make use of their massive parallelism is fulfilled.

The sorting problem is an important problem in computer science. For it, many, both sequential and parallel, algorithms were developed but the time complexity remains at least $O(n\ log(n))$ for the sequential case and $O(log^2 n)$ for the parallel case.

Studying sorting within P system framework is also a challenging task not only because it can produce better results (in some respect) than in the classical sequential case, but also, because we can compute an (ordered) string from an (unordered) multiset of objects. Moreover, one can remark that in the case of cooperative rules (in P systems with symbol-objects and rewriting-like rules) the order of symbols in the left/right hand side of a production does not count. So, we deal with two "types of unorder" and still can compute an "order".

On of the first approaches of sorting with P systems was done in [42] by considering a bead sort algorithm. There, the sorting procedure was constructed on a tissue P system with symport / antiport rules such that the objects were considered beads and the membranes were considered rods. The idea of the algorithm was that beads start to slide in the membrane structure to their appropriate places. The time complexity for this case was linear, which constitute an improvement of the classical sequential sorting algorithm. However, the payoff for this approach was that it requires a number of membranes proportional to $m \times k$, where m is the maximal number from the vector that we would like to sort, and k is the dimension of the vector.

Another study on this topic was done in [66], where the P systems with inhibitors/promoters and symport / antiport rules were used to develop comparators and then to organize them in a sorting network. As in the previous case, the input data was placed in different membranes and the computation starts operating on already dissociated elements. Also, the result was not obtained in a halting configuration but in a stable one, meaning that there are still rules applicable, but their application does not change the string/object contents of the membranes, nor the membrane structure itself. The time

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

complexity for algorithms presented was also linear with respect to the number of components. The number of membranes used in the computation was also proportional to the number of components.

In [32] other methods and algorithms for the sorting problem are proposed by considering many variants of P systems. The feature shared by many algorithms presented is that the input components will be placed in an initial input membrane and the computation will dissociate this input according to the relation order between the multiplicities of components. In this way, we will interpret the sorting as the order of elimination of the objects. The idea behind many algorithms presented will be to consume objects from all components at once and, when one component is exhausted, to trigger a signal meaning that we find the next component that must be eliminated. In other algorithms we developed a comparator, using very weak "ingredients", which can be used practically in any sorting network design. For most of the algorithms the time complexity will be also linear, while for the others it will depend on the biggest multiplicity.

Finally, a sorting algorithm based on existing ranking algorithm was proposed in [189].

The chapter is structured in three main parts corresponding to the type of problems that is solved: the strong sorting, the weak sorting and the ranking. All these concepts will be defined in the Sorting section where we will discuss basic definitions. The P system models considered in the paper are introduced in P system preliminaries section where a brief introduction to the framework is made. The last section is dedicated to conclusions and open problems regarding these issues.

We can remark that proving universality by using matrix or programmed grammars, register machines or other constructions (as they are known in literature), usually involves non-deterministic approaches as well as the use of a "trap symbol" which will guarantee that the system will work forever in the "bad" cases. As opposed, a practical problem solved by an algorithm must be a deterministic process and this is because we would like to receive the answer of the problem in a well defined time. Moreover, usually we would like that the system will stop (or reach an equilibrium state) after finishing the computation. These issues combined with the fact that in a multiset we do not have an order give a hint on the difficulty of the sorting problems. However, we presented the results above in order to give a rough overview of the P systems framework and of their applicability to solving problems.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

7.1.2 Sorting Networks

One parallel model for studying integer sorting problems is based on the comparison network, where more comparison operations can be executed at the same time. This characteristics offer the possibility to construct such networks which sorts k values in sublinear time.

A comparison network is built of wires and comparators. A comparator is a device that have two inputs X and Y, and computes the function

$$(X', Y') = \Big(\min(X, Y), \max(X, Y)\Big).$$

The sorting network consists of input wires, output wires and comparators. In a comparison network wires are responsible for passing information from one comparator to another one. Essentially, such a network contains just comparators, linked together with wires. To some extent, we can consider that input wires and output wires are also nodes (like comparators), but they do not compute anything, just store a value.

Formally, a comparison network is a directed acyclic graph where the nodes are comparators, input nodes, or output nodes, and the directed arcs are wires.

A sorting network is a comparison network which produces as output a monotone sequence for any input sequence. The running time of a comparison network is the time elapsed from the initialization of the input nodes to the time when the values reach the output nodes.

7.1.3 Sorting Definitions and Notations

Let $V = \{\underline{i} \mid 1 \leq i \leq k\}$ be an alphabet. A word over V is denoted by

$$w = \prod_{j=1}^{m} a_j = a_1 a_2 \cdots a_m,$$

$m \in \mathbb{N}$, where $a_j \in V$ for each $1 \leq j \leq m$. Here, the symbol of product represents the concatenation.

Example 7.1.1 $w = \underline{23}\ \underline{9}\ \underline{157}$.

The reason why we denote the alphabet symbols by underlined numbers is that in this way we can distinguish the symbols and also have the implicit order associated with natural numbers.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Let also $ord : V \to \{1, \cdots, k\}$ be a bijective function such that $i = ord(\underline{i})$, $1 \leq i \leq k$. Then $i_j = ord(a_j)$ is an ordinal number of the j-th letter of w and $a_j = \underline{i_j}$.

Let $v = \prod_{j=1}^{k} \underline{j}$ be the "alphabet word", made by concatenating elements of the alphabet V in alphabetical order.

Since in the multiset processing we represent multisets by strings, we will need a few formal language definitions and notations.

Let $w = \prod_{j=1}^{m} a_j$ be a string. We denote by

$$Perm(w) = \{\prod_{j=1}^{m} a_{i_j} \mid 1 \leq i_j \leq m, 1 \leq j \leq m;\ i_j \neq i_l, 1 \leq j, l \leq m\}$$

the set of *permutations* of string w, i.e., the set of all strings that can be obtained from w by changing the order of symbols. We denote by

$$SSub(w) = \{\prod_{j=1}^{l} a_{i_j} \mid 1 \leq i_{j-1} < i_j \leq m, 2 \leq j \leq l;\ 0 \leq l \leq m\}$$

the set of *scattered subwords* of w, i.e., the set of all strings that can be obtained from w by deleting some (0 or more, possibly all) of its symbols, and concatenating the remaining ones, preserving the order.

For example, the permutations of the *alphabet word* v are the strings having exactly one occurrence of each letter of V, in arbitrary order. Also, the scattered subwords of v are the strings consisting of some of the letters of V, in alphabetic order.

Now, let us go back to the initial scope of the paper – the sorting. According to the classical definitions of what integer sorting means we can give an "equivalent" definition adapted to the P systems. In this framework we can sort only the numbers represented by the multiplicities of the objects but not considering the corresponding objects, or we can consider both characteristics. According to this distinction we can define:

Definition 7.1.1 *Let $v = \prod_{j=1}^{k} \underline{j}$ be the alphabet word. The word*

$$w = \prod_{j=1}^{k} a_j \in Perm(v),\ k = card(V) \in \mathbb{N}^+,$$

where $a_j \in V$ such that $M(a_j) \leq M(a_{j+1})$, for each $1 \leq j \leq k-1$, is called the ranking string *of the multiset M.*

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Definition 7.1.2 *The word*

$$w = \prod_{j=1}^{k} \underline{j}^{M(a_j)}$$

is called weak sorting string *of the multiset M if $\prod_{j=1}^{k} a_j$ is the ranking string of M. Also, $M' : V \to \mathbb{N}$ defined as $M'(\underline{j}) = M(a_j)$ is the* weak sorting multiset *of M.*

Remark 7.1.1 *This definition stands for the case when we are interested in sorting the multiplicities of the objects and not having to look at the corresponding objects. In other words we sort only "properties" and not "objects and properties". Practically, the symbols from the initial multiset are considered to be in a complete relation order, and, after performing the computation we will obtain as the result these objects sorted according to the relation order and having multiplicities sorted.*

Definition 7.1.3 *The word*

$$w = \prod_{j=1}^{k} a_j^{M(a_j)}$$

is called strong sorting string *of M if $\prod_{j=1}^{k} a_j$ is the ranking string of M.*

Remark 7.1.2 *In this case, we are interested to have as the output of computation the objects with the same associated multiplicities, present in a string in the increasing order of their multiplicities.*

Example 7.1.2 *For the alphabet $V = \{\underline{1}, \underline{2}, \underline{3}\}$ and the multiset $M = \{(\underline{1}, 20), (\underline{2}, 10), (\underline{3}, 30)\}$, we have:*

- *ranking string :* $\underline{2}\,\underline{1}\,\underline{3}$
- *weak sorting string :* $\underline{1}^{10}\,\underline{2}^{20}\,\underline{3}^{30}$
- *strong sorting string :* $\underline{2}^{10}\,\underline{1}^{20}\,\underline{3}^{30}$

We will typically consider the starting configuration of the sorting P system depending only on the number $k = Card(V)$ of components, taking as the input the multiset

$$\{(\underline{j}, n_j) \mid 1 \leq j \leq k\}$$

over $V \subset O$, placed in a specific region, where O is the system's alphabet.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

7.1.4 Weak Sorting

The weak sorting is an algorithm which processes a multiset M and gives as a result a word $w = \prod_{j=1}^{k} \underline{j}^{M(a_j)}$ such that $\prod_{j=1}^{k} a_j$ is the ranking string of M. A weaker definition could be given: the multiset M' corresponding to w can be considered as the result ($M'(\underline{j}) = M(a_j)$, that is, the objects' multiplicities are ordered) because the order of numbers is represented by the order of the corresponding objects in the alphabet. In the case of equality of some numbers there is no need to make a decision whether " $=$ " is " $\leq$ " or " $\geq$ "; the result is the same. A typical strategy for the weak sorting would be a *sorting network* i.e. a parallel or sequential usage of compare-swap-if-needed operator

$$(n, n') \rightarrow (\min(n, n'), \max(n, n')).$$

The weak sorting is typically easier than the strong sorting.

7.1.5 Evolution–Communication Systems

The idea of the next algorithm is very similar to that of the previous paragraph (a parallel simulation of *odd-even sorting network*). However, the model of the P system is quite different. We now have no target indications in the evolution rules (the result stays in the same regions), and we have communication rules (that do not change objects, just move them). These two types of rules are executed in a maximally parallel manner. In the following construction, we will only have non-cooperative evolution rules, symport of weight 1 and antiport of weight 1.

The *weak* priority can exist between the evolution rules of the same region, between the communication rules of the same membrane, and between the rules of the two classes, where the corresponding membrane is the external bound of the corresponding region.

Example 7.1.3 *Let $B_4 = b_4$, $D_4 = d_4$, i.e., there b_4 and d_4 are used as synonyms of B_4 and D_4. The initial configuration is the following:* $[_1\ [_2\ a^{n_1}b^{n_2}c^{n_3}d^{n_4}\ ]_2\ ]_1$
The rules are defined by the tables below.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Region 1	**Region 2**	$i \in$	*Step*
	$a \to a_0,\ b \to b',$ $c \to c_0,\ d \to d'$		*1*
	$(a_0, out),\ b' \to b_0,$ $(c_0, out),\ d' \to d_0$		*2*
$a_0 \to a_1,\ c_0 \to c_1$	$b_0 \to b_1,\ d_0 \to d_1$		*3*
	$(a_i, in; b_i, out) >$ $\{(a_i, in), b_i \to a_i\},$ $(c_i, in; d_i, out) >$ $\{(c_i, in), d_i \to c_i\}$	$\{1,3\}$ $\{1,3\}$	*4,12*
$b_i \to A_{i+1},\ d_i \to C_{i+1}$	$a_i \to B_{i+1},\ c_i \to D_{i+1}$	$\{1,3\}$	*5,13*

Region 1	**Region 2**	$i \in$	*Step*
	$(A_i, in),\ (B_i, out),$ $(C_i, in),\ (D_i, out)$	$\{2\}$	*6*
$B_i \to b_i,\ D_i \to d_i$	$A_i \to a_i,\ C_i \to c_i$	$\{2\}$	*7*
$d_i \to d_i'$	$(b_i, in; c_i, out) >$ $\{(b_i, in), c_i \to b_i\},$ $a_i \to a_i'$	$\{2\}$ $\{2\}$	*8*
$a_i' \to A_{i+1},\ b_i \to C_{i+1}$	$c_i \to B_{i+1},\ d_i' \to D_{i+1}$	$\{2\}$	*9*
	$(A_i, out),\ (B_i, in),$ $(C_i, out),\ (D_i, in)$	$\{1\}$	*10*
$A_i \to a_i,\ C_i \to c_{i+1}$	$B_i \to b_i,\ D_i \to d_i$	$\{1\}$	*11*
$A_4 \to a_4,\ C_4 \to c_4$	$(B_4, out),\ (D_4, out)$		*14*

Region 1	*Step*
$a_4 \to a,\ b_4 \to b_5,\ c_4 \to c_5,\ d_4 \to d_5$	*15*
$(a, out),\ b_5 \to b,\ c_5 \to c_6,\ d_5 \to d_6$	*16*
$(b, out),\ c_6 \to c,\ d_6 \to d_7$	*17*
$(c, out),\ d_7 \to d$	*18*
(d, out)	*19*

Here, the rules written in the same row of the table are executed in the same step, and the table also indicates the actual step when these rules can be applied. Below is the computation

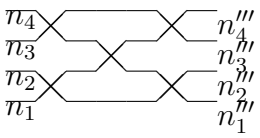
$$
\begin{array}{ll}
[_1\ [_2\ a^{n_1} b^{n_2} c^{n_3} d^{n_4}\ ]_2\ ]_1 \Rightarrow & [_1\ [_2\ a_0^{n_1} b'^{n_2} c_0^{n_3} d'^{n_4}\ ]_2\ ]_1 \Rightarrow \\
[_1\ a_0^{n_1} c_0^{n_3} [_2\ b_0^{n_2} d_0^{n_4}\ ]_2\ ]_1 \Rightarrow & [_1\ a_1^{n_1} c_1^{n_3} [_2\ b_1^{n_2} d_1^{n_4}\ ]_2\ ]_1 \Rightarrow \\
[_1\ b_1^{n_1'} d_1^{n_3'} [_2\ a_1^{n_2'} c_1^{n_4'}\ ]_2\ ]_1 \Rightarrow & [_1\ A_2^{n_1'} C_2^{n_3'} [_2\ B_2^{n_2'} D_2^{n_4'}\ ]_2\ ]_1 \Rightarrow
\end{array}
$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$[_1\ B_2^{n_2'} D_2^{n_4'} [_2\ A_2^{n_1'} C_2^{n_3'}\ ]_2\ ]_1 \Rightarrow$ $[_1\ b_2^{n_2'} d_2^{n_4'} [_2\ a_2^{n_1'} c_2^{n_3'}\ ]_2\ ]_1 \Rightarrow$
$[_1\ c_2^{n_2''} d_2'^{n_4''} [_2\ a_2'^{n_1''} b_2^{n_3''}\ ]_2\ ]_1 \Rightarrow$ $[_1\ B_3^{n_2''} D_3^{n_4''} [_2\ A_3^{n_1''} C_3^{n_3''}\ ]_2\ ]_1 \Rightarrow$
$[_1\ A_3^{n_1''} C_3^{n_3''} [_2\ B_3^{n_2''} D_3^{n_4''}\ ]_2\ ]_1 \Rightarrow$ $[_1\ a_3^{n_1''} c_3^{n_3''} [_2\ b_3^{n_2''} d_3^{n_4''}\ ]_2\ ]_1 \Rightarrow$
$[_1\ b_3^{n_1'''} d_3^{n_3'''} [_2\ a_3^{n_2'''} c_3^{n_4'''}\ ]_2\ ]_1 \Rightarrow$ $[_1\ A_4^{n_1'''} C_4^{n_3'''} [_2\ B_4^{n_2'''} D_4^{n_4'''}\ ]_2\ ]_1 \Rightarrow$
$[_1\ a_4^{n_1'''} B_4^{n_2'''} c_4^{n_3'''} D_4^{n_4'''} [_2\ ]_2\ ]_1 \Rightarrow$ $[_1\ b_5^{n_2'''} c_5^{n_3'''} d_5^{n_4'''} [_2\ ]_2\ ]_1 a^{n_1'''} \Rightarrow$
$[_1\ c_6^{n_3'''} d_6^{n_4'''} [_2\ ]_2\ ]_1 a_4^{n_1'''} b^{n_2'''} \Rightarrow$ $[_1\ d_7^{n_4'''} [_2\ ]_2\ ]_1 a^{n_1'''} b^{n_2'''} c^{n_3'''} \Rightarrow$
$[_1\ [_2\ ]_2\ ]_1 a^{n_1'''} b^{n_2'''} c^{n_3'''} d^{n_4'''}$.

The objects are divided in two classes: $\{\underline{j} \in V \mid j \equiv 0(mod\ 2)\}$ and $\{\underline{j} \in V \mid j \equiv 1(mod\ 2)\}$. These classes are stored in different membranes, so the compare-swap-if-needed operator sorting A, B to A', B' is made of an antiport rule $(A, out; B, in)$, having a weak priority over a symport rule (A, out) and over a rewriting rule $B \to A$, after which A is renamed to $\overline{B}$, B is renamed to $\overline{A}$, both types of objects cross the membrane and then $\overline{B} \to B'$, $\overline{A} \to A'$. A comparison operator is executed in 4 steps.

In this example, the following sorting network $(n_i)_{1 \le i \le 4} \Rightarrow (n_i''')_{1 \le i \le 4}$ was simulated.



$(n_1, n_2) \to (n_1', n_2')$, $(n_3, n_4) \to (n_3', n_4')$,
$n_1'' = n_1'$, $(n_2', n_3') \to (n_2'', n_3'')$, $n_4'' = n_4'$,
$(n_1'', n_2'') \to (n_1''', n_2''')$, $(n_3'', n_4'') \to (n_3''', n_4''')$,
that can be represented like the picture to the left.

The general case of sorting k numbers is done in a similar way. The depth of the sorting network should be $k - ((k+1) mod\ 2)$.

7.1.6 Summary

We have studied the possibility to solve the sorting problem in the P system framework by considering the main variants of membrane devices. The interesting result concerning this topic is that starting with objects that does not have any order and being mixed together in what formally we call a multiset, we constructed the order by computing. We studied in this way many membrane system models which behave in a slightly different ways when they address the same problem. The common feature shared by many algorithms presented is that we sort by "carving" (consuming objects iteratively, one symbol from all the components at once) and signaling when a

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

modification occurs in the system (usually we trigger a signal when a certain component was eliminated). Other ideas were to use the classical approaches of sorting by using a comparator. This comparator was implemented using only input/output operations and catalytic rules. The device presented can be adapted to work in many algorithms that are comparison-based.

Sorting problems are among the most important problems in computer science theory. Beside them there are a lot of other problems which are waiting to be solved in the framework of P systems. The reason is that we can obtain better results in time complexity than the classical algorithms by using the massive parallelism feature of the P systems. We believe that some techniques (the comparison method in the case of movable catalysts and non-cooperative rules, the synchronization methods used in the comparison based algorithms) developed in the paper can be also used to construct systems that solve such kind of problems. The improvements of current algorithms (by reducing the number of membranes when this is the case, reducing the number of catalysts and so on) are also left open. Yet another interesting research topic is designing *dynamic* sorting (universal for arbitrary number of components) P systems.

7.2 Solving NP-Complete Problems

We present an $O(n) + O(\log m)$-time solution of `SAT` with n variables and m clauses by a uniform family of deterministic P systems with communication rules (antiport-2/1 and antiport-1/2) and membrane division rules (without polarization) and unstructured environment. Nothing is sent to the environment except `yes` or `no`, in one copy. We can even start with the empty environment if we also use symport-1 rules.

This section is a continuation of studies of [166], where membrane division rules were added to (tissue) P systems with symport and antiport, to solve `SAT` in a uniform way. Here we improve the results from [166] in the following:

- Tissue P system is replaced by a ("usual") P system with tree-like structure;
- The P system gives the result in time which depends on the number of clauses logarithmically, not linearly;
- The computation is now deterministic, not just confluent;

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- The cardinality of the environment alphabet E is only 1.

- Only antiport-2/1 and antiport-1/2 are used, while the construction from [166] used symport-1, and antiports 1/2, 2/1, 3/2.

- Nothing is sent into environment except the result, just like P systems with active membranes ([164]).

- Paying the price of additionally using symport-1 we no longer need to bring any objects from the environment.

The determinism can be reached due to the massive parallelism (what could happen in either order should happen simultaneously) and the system does not need resources (supply of objects) from the environment because the number of objects can grow via membrane division.

7.2.1 Symport / Antiport and Membrane Division

A P system with symport / antiport and membrane division is defined as a tuple $\Pi = (O, E, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, R, i_0)$, where O is a finite set of objects, $E \subseteq O$ is a set of objects present in the environment in the unbounded quantities, μ is a membrane structure with m regions. w_i, $1 \leq i \leq m$ are the strings representing initial multisets of each region i. R_i, $1 \leq i \leq m$ are the rules associated to each membrane i. The communicative rules are of one of the following forms: (u, in), (v, out), $(u, out; v, in)$, $u, v \in O^+$ (the first two forms are called symport, while the latter is called antiport). The membrane division rules are of the form $[_h\ a\ ]_h \rightarrow [_h\ b\ ]_h [_h\ c\ ]_h$.

The rules are applied non-deterministically, in a maximally parallel manner. The system is deterministic if there is only one computation possible (for any reachable configurations, all configurations reachable in one step are indistinguishable). Notice that this property does not imply that there is only one rule applicable for every object because different copies of the same object are indistinguishable, and so are different membranes with the same label and contents.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

7.2.2 Solving SAT

The problem is defined as follows: given a boolean formula

$$\begin{aligned} \gamma &= C1 \wedge \cdots \wedge C_m, \text{ where} \\ C_i &= y_{i,1} \vee \cdots \vee y_{i,k_i},\ 1 \leq i \leq m, \\ &\quad y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\},\ 1 \leq i \leq m,\ 1 \leq j \leq k_i \end{aligned}$$

find whether γ has a solution.

We define M as $Ceil(\log_2 m)$ $(= \min\{j \in \mathbb{N} \mid 2^j \geq m\})$, this is a number, logarithmic with respect to the number of clauses, which will be needed for defining the system below (notice that $2^M < 2m$). Look at the set S defined below (this is a set of objects that we will need in the environment in a sufficient number of copies to perform the clause evaluation in all membranes in parallel). Notice that $|S| = 6n + 2n + (M+1)n + 2^M + M + 1 = n(M+9) + 2^M + M + 1 = O(nM) + O(m)$. We also define N as $Ceil(\log_2 |S|)$ $(= \min\{j \in \mathbb{N} \mid 2^j \geq |S|\})$, this is a number, logarithmic with respect to $|S|$ (notice that $2^N < 2|S| = O(nM) + O(m)$).

We define T as $6n+2N+2M+3$ (the time we claim is enough to produce object *yes* in the skin membrane if and only if γ is satisfiable). The instance of a problem is encoded in the alphabet $\Sigma = \{s_{i,j}, s'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$ by objects $s_{i,j}$ if clause j contains x_i, and $s'_{i,j}$ if clause j contains $\neg x_i$. We construct the following P system

$$\begin{aligned} \Pi &= (O, E = \{\$\}, \mu, w_0, w_1, w_2, w_3, w_4, R_0, R_1, R_2, R_3, R_4, R), \\ O &= \{\$, f, d, a_1, \texttt{yes}, \texttt{no}\} \cup \{e_i \mid 0 \leq i \leq 2n+N\} \\ &\cup \{d_{i,0} \mid 0 \leq i < n+N\} \cup \{d_{2n+N+i,j} \mid 0 \leq i \leq N, 0 \leq j \leq 2^i\} \\ &\cup \{t'_i, f'_i \mid 1 \leq i \leq n\} \cup \{b_i \mid 0 \leq i \leq T\} \cup \Sigma, \\ S &= \{t''_i, t'''_i, t''''_i, f''_i, f'''_i, f''''_i, \mid 1 \leq i \leq n\} \\ &\cup \{a_i \mid 2 \leq i \leq 2n+1\} \cup \{t_{i,j}, f_{i,j} \mid 1 \leq i \leq n, 0 \leq j \leq M\} \\ &\cup \{c_{2^j(2i),2^j(2i+1)} \mid 0 \leq j \leq M, 0 \leq i < 2^{M-j-1}\} \\ &\cup \{z_j \mid 0 \leq j < M\} \cup \{z\} \text{ are synonyms} \\ &\quad \text{of some symbols } d_{2n+2N,j}, 0 \leq j \leq 2^N, \end{aligned}$$

$$\begin{aligned} \mu &= [_0\,[_1\ ]_1[_2\ ]_2[_3\ ]_3[_4\ ]_4[_5\ ]_5\,]_0, \\ w_0 &= e_{n+N}f,\ w_1 = a_1 d,\ w_2 = \texttt{yes},\ w_3 = b_0\texttt{no},\ w_4 = e_0 e_{n+N},\ w_5 = d_{0,0}z. \end{aligned}$$

The following rules are used:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

- **Cloning objects** e_{n+N}

E1 $[_4\ e_i\ ]_4 \to [_4\ e_{i+1}\ ]_4[_4\ e_{i+1}\ ]_4 \in R$, $0 \leq i < n+N$;

E2 $(e_{n+N}e_{n+N}, out; e_{n+N}, in) \in R_4$;

Starting from $[_4\ e_0e_{n+N}\ ]_4$, in $n+N$ steps, 2^{n+N} copies of $[_4\ e_{n+N}e_{n+N}\ ]_4$ are produced by rule E1. Then, starting from one copy of e_{n+N}, in $n+N$ more steps, 2^{n+N} copies of e_{n+N} are produced by rule E2.

- **Producing objects for the computation**

O1 $[_5\ d_{i,0}\ ]_5 \to [_5\ d_{i+1,0}\ ]_5[_5\ z\ ]_5 \in R$, $0 \leq i < n+N$;

O2 $[_5\ d_{i,0}\ ]_5 \to [_5\ d_{i+1,0}\ ]_5[_5\ d_{i+1,0}\ ]_5 \in R$, $n+N \leq i < 2n+N$;

O3 $[_5\ d_{i,j}\ ]_5 \to [_5\ d_{i+1,2j}\ ]_5[_5\ d_{i+1,2j+1}\ ]_5 \in R$,
$2n+N \leq i < 2n+2N$, $0 \leq j \leq 2^N$;

O4 $(d_{2n+2N,j}z, out; e_n, in)$, $0 \leq j < 2^N$;

In membrane 5, the object $d_{i,0}$ "waits" for $n+N$ steps, i.e., changes into $d_{n+N,0}$ by rules O1 (also $n+N$ dummy membranes are produced). In another n steps, from $[_5\ d_{n+N,0}z\ ]_5$, rules O2 produces 2^n copies of $[_5\ d_{2n+N,0}z\ ]_5$. In further N steps, each of these 2^n membranes divides by rule O3 and produces $[_5\ d_{2n+2N,0}z\ ]_5, \cdots, [_5\ d_{2n+2N,2^N-1}z\ ]_5$, i.e., 2^N membranes with different objects. These are the objects which we will need in the skin membrane for the further computation, they are simultaneously brought in the skin by rule O4 (let us give, for simplicity, pseudonyms to objects $d_{2n+2N,j}$ for different j from entire set S: $d_{2n+2N,0} = t'_1, d_{2n+2N,1} = t'_2, \cdots, d_{2n+2N,|S|} = z$).

- **Variable assignments**

A1 $[_1\ a_i\ ]_1 \to [_1\ t'_i\ ]_1[_1\ f'_i\ ]_1 \in R$, $1 \leq i \leq n$;

A2 $(t'_i, out; t''_i a_{i+1}, in) \in R_1$,
$(f'_i, out; f''_i a_{i+1}, in) \in R_1$, $1 \leq i \leq n$;

A3 $(a_{n+i}t''_i, out; t'''_i, in) \in R_1$,
$(a_{n+i}f''_i, out; f'''_i, in) \in R_1$, $1 \leq i \leq n$;

A4 $(t'''_i, out; t_{i,0}a_{n+i+1}, in) \in R_1$,
$(f'''_i, out; f_{i,0}a_{n+i+1}, in) \in R_1$, $1 \leq i \leq n$;

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

A5 $(t_{i,j}, out; t_{i,j+1}t_{i,j+1}, in) \in R_1$,
$(f_{i,j}, out; f_{i,j+1}f_{i,j+1}, in) \in R_1$, $1 \leq i \leq n$, $0 \leq j < M$.

Using rules A1 and A2, $[_1\ a_i\ ]_1$ changes to $[_1\ t''_{i+1}a_{i+1}\ ]_1[_1\ f''_{i+1}a_{i+1}\ ]_1$ in two steps, except for the case $i = 1$, where objects t'_1 and f'_1 wait for $2n + 2N$ steps until the objects from S are produced and moved to the environment. In $2n$ steps, rules A3 and A4 exchange objects t''_i in membranes labelled 1 for objects $t_{i,0}$ (and f''_i for $f_{i,0}$). After some object $t_{i,0}$ ($f_{i,0}$) appears in membrane 1, it is exchanged for 2^M copies of $t_{i,M}$ ($f_{i,0}$, respectively), this replication takes another $2n$ steps. In this way, there will be enough copies of $t_{i,M}$ (and $f_{i,0}$), i.e., witnesses of true/false assignment of variable x_i for each object $s_{i,j}$ (and $s'_{i,j}$), encoding the fact that if x_i is true (false, respectively), then clause C_j is satisfied. This part of computation will finish in at most $6n + 2N + M$ steps from the beginning of the computation.

- **Checking clauses**

C1 $(t_{i,M}s_{i,j}, out; c_{j-1,j}, in) \in R_1$,
$(f_{i,M}s'_{i,j}, out; c_{j-1,j}, in) \in R_1$, $1 \leq i \leq n$;

C2 $(c_{2^j(2i),2^j(2i+1)}c_{2^j(2i+1),2^j(2i+2)}, out; c_{2^j(2i),2^j(2i+2)}, in) \in R_1$,
$0 \leq j \leq M$, $0 \leq i \leq \frac{m}{2^j \cdot 2} - 1$,
$(c_{2^j(2i),2^j(2i+1)}, out; z_j c_{2^j(2i),2^j(2i+2)}, in) \in R_1$, $0 \leq j < M$, $\frac{m}{2^j \cdot 2} - 1 < i \leq \frac{2^M}{2^j \cdot 2} - 1$;

C3 $(c_{0,2^M}d, out; z, in) \in R_1$;

C4 $(\texttt{yes}, out; fd, in) \in R_2$;

C5 $[_3\ b_i\ ]_3 \to [_3\ z\ ]_3[_3\ b_{i+1}\ ]_3 \in R$, $0 \leq i < T$;

C6 $(b_T\texttt{no}, out; f, in) \in R_3$;

C7 $(\texttt{yes}, out; \$\$, in) \in R_0$, $(\texttt{no}, out; \$\$, in) \in R_0$.

All clauses are checked simultaneously by rules C1: clause C_j satisfied corresponds to object $c_{j-1,j}$. Then rules C2 "assemble" the clause satisfiability: objects c_{j_1,j_2} and c_{j_2,j_3} are replaced by an object c_{j_1,j_3}, but this assembly has a binary character: $j_1 = 2^j(2i)$, $j_2 = 2^j(2i + 1)$, $j_3 = 2^j(2i + 2)$ ($c_{0,1} + c_{1,2} \longrightarrow c_{0,2}$, $c_{2,3} + c_{3,4} \longrightarrow c_{2,4}$, $c_{0,2} + c_{2,4} \longrightarrow c_{0,4}$, etc.) The second

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

group of rules provide "automatic satisfiability" of the non-existing clauses $m+1, \cdots, 2^M$.

In this way in every membrane corresponding to a solution of γ, an object $c_{0,2^M}$ will be obtained. Then it will come in the skin membrane together with an object d using rule C3. Next, d will be exchanged by `yes` by C4, also removing f from the skin (if and only if γ is satisfiable; at most one copy).

This will happen at $M+3$ steps after the end of the variable assignment phase (this number is exact because all clauses need to be checked and contribute to this result), i.e., at step $6n+2N+2M+3=T$. Meanwhile, in membrane 3 a counting until T is performed by rules C5. Rule C6 is executed if and only if the object f is still in the skin (i.e., when γ is not satisfiable).

Then, either `yes` or `no` will be brought into the environment by C7. The system halts in time $T+1$ if the formula is satisfiable, and in time $T+2$ otherwise.

Replacing rules C7 by $(\texttt{yes}, out))$ and $(\texttt{no}, out)$ and redefining E as $\emptyset$ would lead to an equivalent solution.

7.2.3 Summary

The satisfiability problem for a boolean formula in the disjunctive normal form with n variables and m clauses can be solved in time $O(n)+O(\log m)$ by a uniform family of deterministic P systems with communication rules (antiport-2/1, antiport-1/2, symport-1) and membrane division rules (without polarization) and empty environment. The use of symport rules can be eliminated at a price of statring with (at least 2 copies of) one symbol in the environment, and the system will only eject the result in the environment.

We would like to make the following comments:

- The determinism heavily depends on the massive parallelism (not just in different membranes corresponding to different clauses, but also in each membrane for the same clause). Is massive parallelism of rules with respect to membranes (i.e., using for some membrane the number of rules not bounded by a constant independent of n and m) really needed in order to have a deterministic construction, or it is only needed for a construction that runs in logarithmic time with respect to the number of clauses?

- Increasing both the number of membranes and objects heavily depends on using membrane division. Since in the communicative model the

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

objects can be taken from the environment, it would be interesting to consider a variant of membrane systems where membrane division does not increase the number of objects. For instance, can SAT be solved by communicative P systems with membrane separation? (See [23] for the definition of membrane separation)

- We expect that the number of starting membranes can be decreased. Is such a solution possible starting with two membranes?

7.3 From Protons to Bi-stable Catalysts

We will now switch to another kind of applications: applying proton pumping results from Chapter 4 to the studies of maximally parallel multiset rewriting systems, namely, with bi-stable catalysts.

An interesting observation is that, interpreting the same object in different regions of the system as different objects in the same region (encoding regions in objects), one can easily see that the proton becomes a bi-stable catalyst. Let us explain this more formally.

Given a proton pumping P system with two membranes $\Pi = (O, P, [_1\,[_2\,]_2\,]_1, w_1, w_2, R_1, R_2, R'_1, R'_2)$ such that the communication rules are minimally cooperative (either symport rules of weight at most two and antiport rules of weight 1) and the only rules associated to the skin membrane are the rules that output the terminal symbols, one can construct a P system with bi-stable catalysts in the following way:

$$
\begin{aligned}
\Pi' &= (O', C_b, [_1\ \ ]_1, w'_1, R') \text{ where}\\
O' &= \{a, h(a) \mid a \in O\} \cup \{b_p \mid \{p, h(p)\} \in C_b\},\\
C_b &= \{\{p, h(p)\} \mid p \in P\},\\
w'_1 &= h_b(w_1 h(w_2)),\\
R' &= R_1 \cup \{h(u) \to h(v) \mid (u \to v) \in R_2\} \cup R'',\\
R'' &= \{a \to a_{out} \mid (a, out) \in R'_1\} \cup \{h(u) \to u \mid (u, out) \in R'_2\}\\
&\cup\ \{u \to h(u) \mid (u, in) \in R'_2\} \cup \{h(u)v \to h(v)u \mid (u, out; v, in) \in R'_2\}\\
&\cup\ \{h(p)b_p \to pb_p \mid (p, out) \mid R'_2,\ p \in P\}\\
&\cup\ \{pb_p \to h(p)b_p \mid (p, in) \mid R'_2,\ p \in P\},
\end{aligned}
$$

where $h : O \to \{a' \mid a \in O\}$ and $h_b : O \to O^*$ are morphisms defined by $h(a) = a'$ for every $a \in O$, $h_b(a) = a$ for $a \in O - \{p, p' \mid p \in P\}$, $h_b(p) = pb_p$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

for $p \in P$, and $h_b(p') = p'b_p$: h is the priming morphism for objects of region 2, and h_b is the morphism adding objects b_p to objects p or p'.

It is easy too see that the behavior of Π' is exactly the same as that of Π: the objects in region 1 of Π are also in Π', while the objects in region 2 of Π are renamed (i.e., primed) and also placed in region 1 of Π, and the rules are changed accordingly. The role of extra objects b_p (one copy for every copy of bi-catalytic symbols in w_1 and w_2) is to transform all non-cooperative proton rules in cooperative bi-stable catalytic rules (because rules $p \to p'$ or $p' \to p$, $\{p, p'\} \in C_b$, are forbidden by the definition of P system with bi-stable catalysts).

Clearly, non-cooperative rules (except the uniport of protons) remain non-cooperative, while other rules are changed as follows:

In Π	(pa, out)	(pa, in)	$(p, out; a, in)$	$(a, out; p, in)$
In Π'	$p'a' \to pa$	$pa \to p'a'$	$p'a \to pa'$	$pa' \to p'a$

In Π	(p, out)	(p, in)
In Π'	$p'b_p \to pb_p$	$pb_p \to p'b_p$

We can now claim that during this transformation the proton pumping computational completeness constructions become the computational completeness constructions of P systems with (the same number as protons in the original construction) bi-stable catalysts.

Example 7.3.1 *Transformed time-free P system from Corollary 4.8.3 to Theorem 4.3.2 (extra objects are not needed: the construction does not have uniport rules of protons).*

$$\begin{aligned}
\Pi &= (O, C_b, [_1\ \ ]_1, w_1, R_1), \text{ where} \\
O &= \{a_i, a'_i \mid 1 \le i \le m+2\} \cup \{l_j, l'_j \mid l \in I_-, 1 \le j \le 4\} \\
&\cup \{\#_1, \#_2, \#'_1, \#'_2\} \cup \{l, l' \mid l \in I\} \cup P, \\
C_b &= \{\{D_i, D'_i\}, \{E_i, E'_i\} \mid i \in W\}, \\
w_1 &= l_0 \#_1 D'_{m+1} D'_{m+2} Z'_{m+1} Z'_{m+2},
\end{aligned}$$

and the sets of rules are the following:

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$\begin{aligned}
R_1 &= \{l \to (a_i)_{out} l^{(1)}, l \to (a_i)_{out} l^{(2)} \mid l : (A(i), l^{(1)}, l^{(2)}) \in I,\ 1 \le i \le m\} \\
&\cup \{l \to a_i l^{(1)}, l \to a_i l^{(2)} \mid l : (A(i), l^{(1)}, l^{(2)}) \in I, i \in W\} \cup \{\#_2 \to \#_2\} \\
&\cup \{l_4 \to l^{(1)}, l_1 \to l_2, l_2 \to \#_2, l_3 \to l^{(2)} \mid l : (S(i), l^{(1)}, l^{(2)}) \in I\}, \\
&\cup \{l' \to l'_4, l' \to l'_1, l'_2 \to l'_3 \mid l : (S(i), l^{(1)}, l^{(2)}) \in I\} \cup \{\#'_1 \to \#'_1\}, \\
&\cup \{l \to l', l'_4 D'_i \to l_4 D_i, a_i D_i \to a'_i D'_i, \#'_1 D'_i \to \#_1 D_i, l'_1 E'_i \to l_1 E_i, \\
&\quad a_i E_i \to a'_i E'_i, l_2 E_i \to l'_2 E'_i, l'_3 \to l_3 \mid l : (S(i), l^{(1)}, l^{(2)}) \in I\}.
\end{aligned}$$

Thus we obtain a (clearly, optimal) computational completeness result for systems with one bi-stable catalyst: $LOP_1(2cat_1, tar) = RE$, improving $NOP_5(cat_2, 2cat_1, tar) = NRE$ from [135]. Another new result (see the example above) is that time-free systems with four bi-stable catalysts are computationally complete: $fLOP_1(2cat_4, tar) = RE$ (improving $fPsOP_1(2cat_*, tar) = PsRE$ from [61]).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Chapter 8

Conclusions and Open Problems

We will first repeat the key notations, then we will list the results proved in this thesis. Then we will give a number of conclusions and state some open problems.

8.1 List of Key Notations

Behavior of a P system Π

$N(\Pi)$	Set of numbers generated by Π
$Ps(\Pi)$	Set of vectors generated by Π
$L(\Pi)$	Language generated by Π
$N_a(\Pi)$	Set of numbers accepted by Π
$Ps_a(\Pi)$	Set of vectors accepted by Π
$A_I(\Pi)$	Language accepted by a P automaton Π (initial mode)

Behavior of a family F of P systems

NF	Family of sets of numbers generated by P systems $\Pi \in F$
PsF	Family of sets of vectors generated by P systems $\Pi \in F$
LF	Family of languages generated by P systems $\Pi \in F$
N_aF	Family of sets of numbers accepted by P systems $\Pi \in F$
Ps_aF	Family of sets of vectors accepted by P systems $\Pi \in F$
A_IF	Family of languages accepted by P automata $\Pi \in F$

Classes of P systems (with at most m membranes, $m \in \mathbb{N}$).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

OP_m	P systems with symbol objects
E_1OP_m	P systems with unstructured environment
$ProP_m^k$	Proton pumping P systems (with at most k protons)
O_nP_m	P systems (with at most n objects)
OtP_m	Tissue P systems with symbol objects
O_ntP_m	Tissue P systems with at most n objects
$O_nt'P_m$	as above, with multiple channels

Features of P systems

$ncoo$	Simple non-cooperative rewriting rules
$ncoo_2$	Simple binary non-cooperative rewriting rules
tar	with targets
p_1ncoo	Simple non-cooperative rules with promoters of weight one
cat_k	Rewriting rules with (at most k in the system) catalysts
$2cat_k$	Rules with (at most k in the system) bi-stable catalysts
sym_i	Symport rules of weight at most i
$sym_{=i}$	Symport rules of weight i
$anti_j$	Antiport rules of weight at most j

In case rewriting rules are absent, an infinite environment is assumed (this is not reflected in the notation).

Properties of P systems

D	Deterministic
f	Time-free

Particular notations can be obtained by combining the elements of these tables: the behavior B of a family F of P systems with features γ satisfying a property Q is denoted by $QBF(\gamma)$.

8.2 List of Results

We now list the (main) results proved in this thesis; numbers are assigned to them for possible references.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Evolution–communication and proton pumping

$$LOP_2(ncoo, sym_1, anti_1) = RE, \tag{8.1}$$
$$LOP_2(ncoo, sym_2) = RE, \tag{8.2}$$
$$A_IOP_2(p_1ncoo, sym_1, anti_1) = RE, \tag{8.3}$$
$$A_IOP_2(p_1ncoo, sym_2) = RE, \tag{8.4}$$
$$LE_1OP_2(ncoo_2, anti_1) = RE, \tag{8.5}$$
$$LE_1OP_2(ncoo_2, sym_{=2}) = RE, \tag{8.6}$$
$$fPsOP_2(ncoo, sym_1, anti_1) = PsRE, \tag{8.7}$$
$$fPsOP_2(ncoo, sym_2) = PsRE, \tag{8.8}$$
$$LOP_2(ncoo, tar, sym_1, anti_1) = RE, \tag{8.9}$$
$$LOP_2(ncoo, tar, sym_2) = RE, \tag{8.10}$$
$$DN_aOP_3(ncoo, sym_1, anti_1) = NRE, \tag{8.11}$$
$$DN_aOP_3(ncoo, sym_2) = NRE, \tag{8.12}$$
$$fLProP_2^4(ncoo, tar, sym_1, anti_1) = RE, \tag{8.13}$$
$$fLProP_2^4(ncoo, tar, sym_2) = RE, \tag{8.14}$$
$$LProP_2^1(ncoo, sym_1, anti_1) = RE, \tag{8.15}$$
$$LProP_2^1(ncoo, sym_2) = RE. \tag{8.16}$$

Symport / antiport of small weight

$$NOP_3(sym_1, anti_1) = NRE, \tag{8.17}$$
$$NOP_3(sym_2) = NRE, \tag{8.18}$$
$$NOtP_2(sym_1, anti_1) = NRE, \tag{8.19}$$
$$DN_aOtP_2(sym_1, anti_1) = NRE, \tag{8.20}$$
$$NOtP_2(sym_2) = NRE, \tag{8.21}$$
$$DN_aOtP_2(sym_2) = NRE, \tag{8.22}$$
$$N_3OP_2(sym_1, anti_1) = N_3RE, \tag{8.23}$$
$$N_6OP_2(sym_2) = N_6RE, \tag{8.24}$$
$$PsOP_2(sym_1, anti_1)_T = PsRE, \tag{8.25}$$
$$PsOP_2(sym_2)_T = PsRE, \tag{8.26}$$
$$NOP_1(sym_1, anti_1) \subseteq NFIN, \tag{8.27}$$
$$NOtP_1(sym_1, anti_1) \subseteq NFIN, \tag{8.28}$$

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

$$NOP_1(sym_1, anti_1) \supseteq SEG_1, \tag{8.29}$$
$$NOP_1(sym_2) \supseteq SEG_1 \cup SEG_2, \tag{8.30}$$
$$N_7OP_1(sym_3) = N_7RE, \tag{8.31}$$

Symport / antiport with a small alphabet

$$Ps(k)O_sP_m(sym_*, anti_*) = Ps_a(k)O_sP_m(sym_*, anti_*) = Ps(k)RE$$
$$\text{for } k = \max\{m(s-2), (m-1)(s-1)\} - 2 > 0, \tag{8.32}$$
$$NO_5P_1(sym_*, anti_*) = N_aO_5P_1(sym_*, anti_*) = NRE, \tag{8.33}$$
$$NO_4P_2(sym_*, anti_*) = N_aO_4P_2(sym_*, anti_*) = NRE, \tag{8.34}$$
$$NO_3P_3(sym_*, anti_*) = N_aO_3P_3(sym_*, anti_*) = NRE, \tag{8.35}$$
$$NO_2P_4(sym_*, anti_*) = N_aO_2P_4(sym_*, anti_*) = NRE, \tag{8.36}$$
$$NO_1P_2(sym_*, anti_*) \supseteq NREG, \tag{8.37}$$
$$NO_2P_1(sym_*, anti_*) \supseteq NREG, \tag{8.38}$$
$$NO_1P_1(sym_*, anti_*) = NFIN, \tag{8.39}$$
$$NO_5t'P_1(sym_*, anti_*) = NRE, \tag{8.40}$$
$$NO_3tP_2(sym_*, anti_*) = NO_3t'P_2(sym_*, anti_*) = NRE, \tag{8.41}$$
$$NO_2tP_4 = NO_2t'P_3(sym_*, anti_*) = NRE, \tag{8.42}$$
$$NO_stP_1(sym_*, anti_*) = NREG,\ s > 1, \tag{8.43}$$
$$NO_2t'P_1(sym_*, anti_*) \supseteq NREG, \tag{8.44}$$
$$NO_1tP_3(sym_*, anti_*) \supseteq NREG, \tag{8.45}$$
$$NO_1t'P_2(sym_*, anti_*) \supseteq NREG, \tag{8.46}$$
$$NO_1tP_1(sym_*, anti_*) = NO_1t'P_1(sym_*, anti_*) = NFIN. \tag{8.47}$$

Applications

A k-dimensional vector of non-negative integers can be sorted (by an evolution–communication P system with priorities of polynomial size with respect to k) in a linear number of steps with respect to k. Actually, the size and time can be bounded linearly with respect to the size and depth of any k-number sorting network. (8.48)

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

An instance of SAT with n variables and m clauses can be solved by a uniform family of deterministic P systems with antiport 2/1 and antiport 1/2 in $O(n \log m)$ steps (only sending the answer object in the environment). (8.49)

$$LOP_1(2cat_1, tar) = RE, \tag{8.50}$$

$$fLOP_1(2cat_4, tar) = RE. \tag{8.51}$$

8.3 List of Conclusions, Open Problems and Research Directions

In this thesis, the studies of membrane systems with rules communicating objects across membranes were presented. We will repeat the topics investigated: multiset rewriting, EC P systems, protons and bi-stable catalysts, symport / antiport of small weight, symport / antiport with a small alphabet, and solving particular problems by P systems.

8.3.1 Multiset Rewriting

It is possible to express most of the interactions between elements (agents, membranes) of P systems via cooperation or context (promoters, inhibitors) in a multiset-rewriting system. On the other hand, we have expressed the notion of maximal parallelism in cooperative multiset-rewriting systems with context via an integer multi-criterial linear optimization problem.

The key element of computing the behavior of a maximally parallel multiset-rewriting system is computing the set of all maximal multisets of applicable rules. The presented algorithm has complexity $O((n/w)^r)$, where n is the number of objects in the system, w is the minimal weight of a multiset in the left-hand side of a rule, and r is the number of rules.

Although the algorithm is quite efficient in some cases, it is a **research direction** to look for more efficient algorithms. For instance, it seems promising to try to express the solution via the solution of a corresponding *continuous* multi-criterial linear optimization problem (its complexity does not depend on n).

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

8.3.2 EC P Systems

We stated a series of investigations about evolution–communication P systems. Since P systems with symport / antiport rules using at most three objects are computationally complete with just one membrane, even without evolution rules, when the infinite environment is present, we only considered rules using at most two objects. The two cases of rules moving two objects are $sym_{=2}$ and $anti_1$. Hence, the two main sets of features we have considered are $\alpha = (ncoo, sym_1, anti_1)$ and $\beta = (ncoo, sym_2)$.

Only two membranes are enough for computational completeness of P systems with features α or β, even in the sense of generating languages. The computational completeness of time-free systems holds in the sense of generating vector sets, or in the sense of languages if targets are allowed, again for both sets of features, with just two membranes. On the other side, the presence of an unstructured environment (and ignoring the corresponding object in the result) allows us to restrict the features to $(ncoo_2, sym_{=2})$ or $(ncoo_2, anti_1)$ and still have computationally complete systems.

It is possible to accept any recursively enumerable language by EC P automata, paying the price of using promoters at the level of non-cooperative rewriting rules. However, considering internal input leads to the following results: deterministic P systems with features α or β are computationally complete with three membranes, in the sense of accepting vector sets.

Intuitively it seems (conjecture) that rewriting in two regions is necessary for universality of EC P systems, and that rewriting in three regions is necessary for universality of deterministic EC P systems. It is an **open** topic to investigate the exact power of evolution–communication P systems with one membrane, as well as deterministic P systems with one or two membranes. An extended list of questions can be found in the preliminary version of [5].

8.3.3 Protons and Bi-stable Catalysts

A special restriction of EC P systems was presented: proton pumping systems. The main focus of this restriction is the way the two objects interact. Surprisingly, this restriction does not ruin the computational completeness of two-membrane systems with features $\alpha = (ncoo, sym_1, anti_1)$ or $\beta = (ncoo, sym_2)$. Four protons ("agents of cooperation") are sufficient for the computational completeness (in the sense of generating languages) of time-free systems, while one proton is already sufficient if time-freeness is

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

not required. We recall that the last fact has been proved by a construction with only four cooperative rules.

P systems with two membranes and k protons can be simulated by P systems with one membrane and k bi-stable catalysts. Hence, the proton results can be transferred to bi-stable catalysts: P systems with one bi-stable catalyst are computationally complete in the sense of generating languages, as well as time-free P systems with four bi-stable catalysts.

It seems that a 3-stable catalyst is needed for the computational completeness of accepting P systems, and that three membranes are needed for the universality of accepting proton pumping P systems with one proton. Completing the picture in this direction is an interesting **research direction**. A few open questions can be found in [6], e.g., it is an **open** question whether four protons/bi-stable catalysts are necessary for time-freeness.

8.3.4 Symport / Antiport of Small Weight

We reflected a series of investigations about purely communicative P systems with rules associated to membranes, focusing on the rules of small weight. We recall that the environment is assumed to contain an infinite supply of objects from a fixed subset of the alphabet. It is known that P systems with rules that move across a membrane one object in one direction and two objects in the other direction are computationally complete as deterministic acceptors (or as generators, with one superfluous object, which can be removed by a symport rule of weight one), while P systems with rules moving up to three objects across a membrane in the same direction are also computationally complete as deterministic acceptors.

We have shown that P systems with symport of weight at most three are computationally complete as generators, with seven superfluous objects (it is an **open** question whether this number can be decreased). In what follows, by "small weight" we will understand two minimal cooperation variants, i.e., either $\alpha = (sym_1, anti_1)$ or $\beta = (sym_2)$.

We have shown that both variants of P systems with three membranes are computationally complete in the sense of generating vector sets, without superfluous objects. In the tissue case, two cells are already enough for both variants; moreover, the result also holds for deterministic acceptors. The results hold, with the same constructions, even if sequential channels are replaced by maximally parallel channels.

Returning to the membrane case, even P systems with two membranes

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

have been shown to be computationally complete as generators of vector sets, but with three superfluous objects with features α or with six superfluous objects with features β (it is an **open** whether these numbers can be decreased).

Finally, we have stated the results on one-membrane systems with minimal cooperation; the power of such systems is very limited (at most finite as generators), but it is still an **open** problem to find their characterizations.

8.3.5 Symport / Antiport with a Small Alphabet

While one "corner" of descriptional complexity of P systems with symport / antiport is the weight of the rules and the number of membranes, another "corner" is the number of objects in the alphabet and the number of membranes.

We presented results on P systems with symport / antiport rules of "heavy" weights and a small alphabet. P systems with $m \geq 1$ membranes and $s \geq 2$ objects are computationally complete as generators or acceptors of number sets when $m+s \geq 6$. A number of results for the other classes was also stated, see the "complexity carpet" in Figure 6.1. The most interesting **open** question is whether P systems with one symbol are computationally complete (it is known that they can simulate partially blind counter automata). It seems (conjecture) that the answer is negative: a membrane seems to need multiple ways of access (multiple symbols or multiple channels).

For tissue P systems, the corresponding question is known to have a positive answer. We have presented results on tissue P systems with symport / antiport rules of "heavy" weights and a small alphabet. Below are the "known frontier" results: The computational completeness holds for tissue P systems with $m \geq 1$ cells and $s > 1$ symbols for $(s, m) \in \{(3, 2), (2, 4), (1, 7)\}$ as generators of number sets. If multiple channels are allowed, then tissue P systems with $m \geq 1$ cells and $s > 1$ symbols are computationally complete as generators of number sets for $(s, m) \in \{(5, 1), (3, 2), (2, 3), (1, 6)\}$. A number of results for the other classes was stated, see the "complexity carpet" in Figure 6.2 and Figure 6.3.

One particular **open** question is whether $NO_1tP_2(sym_*, anti_*)$ can generate $NREG$. It is a **research direction** to research the power of tissue P systems with symport / antiport rules as generators of vector sets, or as acceptors of sets of numbers or vectors, in a way similar to the way cell-like

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

P systems have been investigated.

8.3.6 Solving Particular Problems

While a very restricted set of features is often enough to obtain a universal computational device (often implying that any kind of computational problem can be solved using it), applications of P systems in other areas of science often deal with a much richer set of features. This is often due to the particular way a particular computational problem needs to be solved (imagine, e.g., modern software implemented on a Turing machine instead of a random access machine, with a polynomial slowdown).

In such a way, certain computation time is one demand, another one would be the adequate (transparent) internal representation of the problem (initial data, final answer, and intermediate stages). We will not talk here about the latter (relevant for modelling biological processes, modelling linguistic phenomena, etc.), and focus on the first one instead. To devise time-efficient algorithms (i.e., P system) for particular problems, massive parallelism (a built-in capability of P systems) needs to be exploited.

We have presented EC P systems solving the sorting problem by processing different pairs of numbers in parallel. Notice that in P systems with symbol object string structures do not exist (unless many membranes are used for this purpose), so all data have to be represented by multisets, in this case - in unary. For a number (i.e., all occurrences of the same symbol in the same region) to be processed (e.g., compared) in a constant time, additional control is needed. In this case we have used priorities. Therefore, two levels of parallelism are used: processing different symbols and processing different copies of the same symbol. The number of steps needed for a P system to solve the sorting problem is linear with respect to the depth of an underlying sorting network.

We have stated the construction of P systems with symport / antiport with membrane division solving `SAT`. Membrane division is a powerful feature that lets the system increase the workspace in a fast way. We also "abuse" the membrane division rules to increase the total number of objects in the system and to do the work of renaming (i.e., unary non-cooperative) rules. Antiport rules perform all necessary interactions between the information stored in the objects.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

List of Figures

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

List of Tables

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Bibliography

[1] B. Alberts et. al., *Essential Cell Biology. An Introduction to the Molecular Biology of the Cell*, Garland Publ, New York, London, 1998.

[2] A. Alhazov, A Note on P Systems with Activators, in [167], 16–19.

[3] A. Alhazov, Generating Classes of Languages by P Systems and Other Devices, in [63], 18–22.

[4] A. Alhazov, Maximally Parallel Multiset-Rewriting Systems: Browsing the Configurations, in [108], 1–10, and *Grammars*, submitted, 2004.

[5] A. Alhazov, Minimizing Evolution–Communication P Systems and EC P Automata, in [63], 23–31, and *New Generation Computing* **22**, 4, 2004, 299–310.

[6] A. Alhazov, Number of Protons/Bi-stable Catalysts and Membranes in P Systems. Time-Freeness, in [86], 102–122, and in [98], 80–96.

[7] A. Alhazov, On the Power of Deterministic EC P Systems, in [167], 11-15, and *Journal of Universal Computer Science* **10**, 5, 2004, 502–508.

[8] A. Alhazov, P Systems Without Multiplicities of Symbol-Objects, *Information Processing Letters*, submitted, 2004, accepted, 2005.

[9] A. Alhazov, Solving SAT by Symport / Antiport P Systems with Membrane Division, in [103], 1–6.

[10] A. Alhazov, M. Cavaliere, Evolution–Communication P Systems: Time-freeness, in [108], 11–18.

[11] A. Alhazov, M. Cavaliere, Proton Pumping P Systems, in [27], 1–16, and [139], 1–18.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[12] A. Alhazov, R. Freund, On Efficiency of P Systems with Active Membranes and Two Polarizations, in [146], 81–94, and in [145], 146–160.

[13] A. Alhazov, R. Freund, P Systems with One Membrane and Symport / Antiport Rules of Five Symbols are Computationally Complete, in [108], 19–28.

[14] A. Alhazov, R. Freund, A. Leporati, M. Oswald, C. Zandron, (Tissue) P Systems with Unit Rules and Energy Assigned to Membranes, *Fundamenta Informaticae*, to appear.

[15] A. Alhazov, R. Freund, M. Oswald, Cell / Symbol Complexity of Tissue P Systems with Symport / Antiport Rules, *International Journal of Foundations of Computer Science* **17**, 1, 2006, 3–26.

[16] A. Alhazov, R. Freund, M. Oswald, Symbol/Membrane Complexity of Symport / Antiport P Systems, in [86], 123–146, and in [98], 97–114.

[17] A. Alhazov, R. Freund, M. Oswald, Tissue P Systems with Antiport Rules and Small Number of Symbols and Cells, in [103], 7–22, and *Developments in Language Theory*, 9th International Conference, DLT 2005, Palermo (C. de Felice, A. Restivo, eds.), *Lecture Notes in Computer Science* **3572**, 2005, 100–111.

[18] A. Alhazov, R. Freund, Gh. Păun, Computational Completeness of P Systems with Active Membranes and Two Polarizations, *Machines, Computations, and Universality*, International Conference, MCU 2004, Saint Petersburg, 2004, Revised Selected Papers (M. Margenstern, ed.), *Lecture Notes in Computer Science* **3354**, Springer, 2005, 82–92.

[19] A. Alhazov, R. Freund, Gh. Păun, P Systems with Active Membranes and Two Polarizations, in [167], 20–36.

[20] A. Alhazov, R. Freund, Agustín Riscos-Núñez, One and Two Polarizations, Membrane Creation and Objects Complexity in P Systems, in [70], 9–18, and *IEEE Computer Press*, to appear.

[21] A. Alhazov, R. Freund, Yu. Rogozhin: Computational Power of Symport / Antiport: History, Advances and Open Problems, in [86], 44–78, and in [98], 1–31.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[22] A. Alhazov, R. Freund, Yu. Rogozhin, Some Optimal Results on Symport / Antiport P Systems with Minimal Cooperation, in [103], 23–36.

[23] A. Alhazov, T.-O. Ishdorj, Membrane Operations in P Systems with Active Membranes, in [167], 37–44.

[24] A. Alhazov, M. Margenstern, V. Rogozhin, Yu. Rogozhin, S. Verlan, Communicative P Systems with Minimal Cooperation, in [145], 162–178.

[25] A. Alhazov, C. Martín-Vide, L. Pan, Solving a PSPACE-Complete Problem by P Systems with Restricted Active Membranes, *Fundamenta Informaticae* **58**, 2, 2003, 67–77.

[26] A. Alhazov, C. Martín-Vide, L. Pan, Solving Graph Problems by P Systems with Restricted Elementary Active Membranes, *Aspects of Molecular Computing* - Essays dedicated to Tom Head on the occasion of his 70th birthday (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), *Lecture Notes in Computer Science* **2950** Festschrift, Springer, 2004, 1–22.

[27] A. Alhazov, C. Martín-Vide, Gh. Păun (Eds.), *Preproceedings of the Workshop on Membrane Computing*, Tarragona, GRLMC Report **28/03**, Rovira i Virgili University, 2003.

[28] A. Alhazov, L. Pan, Polarizationless P Systems with Active Membranes, *Grammars* **7**, 2004, 141–159.

[29] A. Alhazov, L. Pan, Gh. Păun, Trading Polarizations for Labels in P Systems with Active Membranes, *Acta Informaticae* 41, 2-3, 2004, 111–144.

[30] A. Alhazov, Yu. Rogozhin, Minimal Cooperation in Symport / Antiport P Systems with One Membrane, in [108], 29–34.

[31] A. Alhazov, Yu. Rogozhin, S. Verlan, Symport / Antiport Tissue P Systems with Minimal Cooperation, in [103], 37–52.

[32] A. Alhazov, D. Sburlan, Static Sorting Algorithms for P Systems, in [27], 17–40.

[33] A. Alhazov, D. Sburlan: Static Sorting P Systems, in [71], 215–252.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[34] A. Alhazov, D. Sburlan, (Ultimately Confluent) Parallel Multiset-Rewriting Systems with Context, in [167], 45–52, and in [146], 95–103.

[35] A. Alhazov, D. Sburlan, Ultimately Confluent Rewriting Systems. Parallel Multiset-Rewriting with Permitting or Forbidding Contexts, in [145], 178–189.

[36] A. Alhazov, S. Verlan, Sevilla Carpets of Deterministic Non-cooperative P Systems, in [103], 53–60.

[37] O. Andrei, G. Ciobanu, D. Lucanu, Rewriting P Systems in Maude, in [146], 104–118.

[38] I.I. Ardelean, M. Cavaliere, Modelling Biological Processes by Using a Probabilistic P System Software, *Natural Computing* **2**, 2, 2003, 173–197.

[39] F. Arroyo Montoro, *Structures and Biolanguage to Simulate Membrane Computing*, **PhD Thesis**, Univeridad Politecnica de Madrid, Madrid, Spain, 2004.

[40] F. Arroyo, A.V. Baranda, J. Castellanos, C. Luengo, L.F. Mingo, A Recursive Algorithm for Describing Evolution in Transition P Systems, *Pre-Proceedings of Workshop on Membrane Computing*, Curtea de Argeş, 2001, GRLMC Report 17/01, Rovira i Virgili University, Tarragona, 2001, 19–30.

[41] F. Arroyo, C. Luengo, A.V. Baranda, L.F. de Mingo, A Software Simulation of Transition P Systems in Haskell, in [174], 29–44, and in [170], 19–32.

[42] J.J. Arulanandham, Implementing Bead-Sort with P Systems, *Unconventional Models of Computation 2002* (C.S. Calude, M.J. Dinneen, F. Peper, eds.), Lecture Notes in Computer Science **2509**, Springer-Verlag, Heidelberg, 2002, 115-125.

[43] D. Balbotin Noval, M.J. Pérez Jiménez, F. Sancho Caparrini, A MzScheme Implementation of Transition P systems, in [174], 62–80, and in [170], 58–73.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[44] A.V. Baranda, J. Castellanos, F. Arroyo, R. Gonzalo, Towards an Electronic Implementation of Membrane Computing: A Formal Description of Nondeterministic Evolution in Transition P Systems, in [130], 350–359.

[45] F. Bernardini, M. Gheorghe: On the Power of Minimal Symport / Antiport, in [27], 72–83.

[46] F. Bernardini, V. Manca, P Systems with Boundary Rules, in [174], 97–102, and in [170], 107–118.

[47] F. Bernardini, A. Păun, Universality of Minimal Symport / Antiport: Five Membranes Suffice, in [139], 43–45.

[48] D. Besozzi, *Computational and Modelling Power of P Systems*, **PhD Thesis**, Universitá degli Studi di Milano, Italy, 2004.

[49] A. Binder, R. Freund, G. Lojka, M. Oswald, Implementation of Catalytic P Systems, *Implementation and Application of Automata*, 9th International Conference, CIAA 2004, Kingston, Revised Selected Papers (M. Domaratzki, A. Okhotin, K. Salomaa, S. Yu, eds.), Kingston, 2004, 24–33.

[50] C. Bonchis, G. Ciobanu, C. Isbasha, D. Petcu, A Web-based P System Simulator and Its Parallelization, *Unconventional Computation*, 4th International Conference, UC 2005, Sevilla (C. Calude, M.J. Dinneen, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, eds.), *Lecture Notes in Computer Science* **3699**, Springer, 2005, 58–69.

[51] C. Bonchis, C. Isbasa, D. Petcu, G. Ciobanu, WebPS: A Web-based P System Simulator with Query Facilities, in [108], 63–72.

[52] P. Bottoni, C. Martín-Vide, Gh. Păun, G. Rozenberg, Membrane Systems with Promoters/Inhibitors, *Acta Informatica*, **38**, 10, 2002, 695–720.

[53] C.S. Calude, E. Calude, M.J. Dinneen (Eds.), *Developments in Language Theory*, 8th International Conference, DLT 2004, Auckland, *Lecture Notes in Computer Science* **3340**, Springer, Berlin, 2004.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[54] C.S. Calude, M.J. Dinneen, Gh. Păun (Eds.), *Pre-proceedings of Workshop on Multiset Processing*, Curtea de Argeş, 2000, *CDMTCS research report* **140**, Univ. Auckland, 2000.

[55] C.S. Calude, Gh. Păun, Bio-steps beyond Turing, *CDMTCS research report* **226**, Auckland Univ., 2003.

[56] C.S. Calude, Gh. Păun, Computing with Cells and Atoms: After Five Years, *CDMTCS research report* **246**, Univ. of Auckland, 2004.

[57] M. Cavaliere, Evolution, Communication and Observation. From Biology to Membrane Systems and Back, *RNGC TR* **03/2004**, Sevilla University.

[58] M. Cavaliere, *Evolution, Communication, Observation: From Biology to Membrane Computing and Back*, **PhD Thesis**, University of Sevilla, Spain, 2006.

[59] M. Cavaliere, Evolution–Communication P Systems, in [170], 134–145.

[60] M. Cavaliere, Towards Asynchronous P Systems, in [146], 161–173.

[61] M. Cavaliere, V. Deufemia, Further Results on Time-Free P Systems, in [103], 95–116.

[62] M. Cavaliere, R. Freund, A. Leitsch, Gh. Păun, Event-related Outputs of Computations in P Systems, in [108], 107–122.

[63] M. Cavaliere, C. Martín-Vide, Gh. Păun (Eds.), *Brainstorming Week on Membrane Computing*, Tarragona, GRLMC Report **26/03**, Rovira i Virgili University, 2003.

[64] M. Cavaliere, D. Sburlan, Time and Synchronization in Membrane Systems, *Fundamenta Informaticae* **64**, 1-4, 2005.

[65] M. Cavaliere, D. Sburlan, Time-independent P systems, Membrane Computing, in [145].

[66] R. Ceterchi, C. Martín-Vide, P Systems with Communication for Static Sorting, in [63], 101–117.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[67] G. Ciobanu, V.M. Gontineac, Algebraic and Coalgebraic Aspects of Membrane Computing, in [86], 289–311.

[68] G. Ciobanu, M. Gontineac, Mealy Membrane Automata and P Systems Complexity, in [103], 149–164.

[69] G. Ciobanu, D. Paraschiv, Membrane Software. A P System Simulator, *Fundamenta Informaticae* **49**, 1-3, 2002, 61–66.

[70] G. Ciobanu, Gh. Păun (Eds), *Pre-proceedings of the First International Workshop on Theory and Applications of P Systems*, Timişoara, Technical Report **05-11**, Institute e-Austria, Timişoara, 2005.

[71] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.), *Applications of Membrane Computing*, Natural Computing Series, Springer-Verlag, Berlin, 2005.

[72] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, P transducers, *New Generation Computing* **24**, 1, 2006, 1–28.

[73] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, Sevilla Carpets Associated with P Systems, in [63], 135–140.

[74] G. Ciobanu, G. Wenyuan, A Parallel Implementation of Transition P Systems, in [27], 169–184.

[75] A. Cordon-Franco, M.A. Gutierrez-Naranjo, M.J. Pérez-Jiménez, F. Sancho-Caparrini, A Prolog Simulator for Deterministic P Systems with Active Membranes, in [63] 141–154.

[76] E. Csuhaj-Varjú, Gy. Vaszil, New Results and Research Directions Concerning P Automata, Accepting P Systems with Communication Only, in [63], 171–179.

[77] E. Csuhaj-Varjú, G. Vaszil, P automata or Purely Communicating Accepting P Systems, in [170], 219–233.

[78] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[79] M. Davis, E. Weyuker, *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, Academic Press, New York, 1994.

[80] R. Freund, Asynchronous P Systems, in [146], 12–28.

[81] R. Freund, Energy-controlled P systems, in [174], 221–236, and in [170], 247–260.

[82] R. Freund, Generalized P Systems, *Fundamentals of Computation Theory*, FCT'99, Iaşi, (G. Ciobanu, Gh. Paun, eds.), Lecture Notes in Computer Science **1684**, Springer, 1999, 281–292.

[83] R. Freund, Special Variants of P Systems Inducing an Infinite Hierarchy with respect to the Number of Membranes, *Bull. EATCS* **75**, 2001, 209–219.

[84] R. Freund, O.H. Ibarra, Gh. Păun, H.-C. Yen, Matrix Languages, Register Machines, Vector Addition Systems, in [108], 155–168.

[85] R. Freund, L. Kari, M. Oswald, P. Sosík, Computationally Universal P Systems without Priorities: Two Catalysts Are Sufficient, *Theoretical Computer Science* **330**, 2, 2005, 251–266.

[86] R. Freund, G. Lojka, M. Oswald, Gh. Păun (Eds.), *Pre-proceedings of the Sixth Workshop on Membrane Computing* (WMC6), Vienna, Austria, 2005.

[87] R. Freund, C. Martín-Vide, A. Obtułowicz, Gh. Păun, On Three Classes of Automata-Like P Systems, *Developments in Language Theory*, 7th International Conference, DLT 2003, Szeged (Z. Ésik, Z. Fülöp, eds.), *Lecture Notes in Computer Science* **2710**, 2003.

[88] R. Freund, M. Oswald, A Short Note on Analysing P Systems with Antiport Rules, *Bulletin of the European Association for Theoretical Computer Science* **78**, 2002, 231–236.

[89] R. Freund, M. Oswald, GP Systems with Activated/Prohibited Membrane Channels, in [170], 261–269.

[90] R. Freund, M. Oswald, GP Systems with Forbidding Context, *Fundamenta Informaticae* **49**, 1–3, 2002, 81–102.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[91] R. Freund, M. Oswald, P Colonies Working in the Maximally Parallel and in the Sequential Mode, in [70], 49–56.

[92] R. Freund, M. Oswald, P Systems with Activated/Prohibited Membrane Channels, in [170], 261–268.

[93] R. Freund, M. Oswald, P systems with Conditional Communication Rules Assigned to Membranes, in [27], 231–240.

[94] R. Freund, M. Oswald, Tissue P Systems with Symport / Antiport Rules of One Symbol are Computationally Universal, in [103], 187–200.

[95] R. Freund, A. Păun, Membrane Systems with Symport / Antiport: Universality Results, in [170], 270–287.

[96] R. Freund, Gh. Păun, On Deterministic P Systems, submitted, 2003, see [205].

[97] R. Freund, Gh. Păun, M.J. Pérez-Jiménez, Tissue-like P Systems with Channel States, in [167], 206–223 and *Theoretical Computer Science* **330**, 2005, 101–116.

[98] R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa (Eds), *Membrane Computing*. International Workshop, WMC6, Vienna, 2005. Revised Selected and Invited Papers, *Lecture Notes in Computer Science* **3850**, Springer, Berlin, 2006.

[99] P. Frisco, About P Systems with Symport / Antiport, in [167], 224–236.

[100] P. Frisco, H.J. Hoogeboom, P Systems with Symport / Antiport Simulating Counter Automata, *Acta Informatica* **41**, 2–3, 2004, 145–170.

[101] P. Frisco, H.J. Hoogeboom, Simulating Counter Automata by P Systems with Symport / Antiport, in [174], 237–248, and in [170], 288–301.

[102] S. Greibach: Remarks on Blind and Partially Blind One-Way Multicounter Machines, *Theoretical Computer Science* **7**, 1978, 311–324.

[103] M.A. Gutiérrez-Naranjo, Gh. Păun, M.J. Pérez-Jiménez (Eds.), *Cellular Computing (Complexity Aspects)*, ESF PESC Exploratory Workshop, Fénix Editorial, Sevilla, 2005.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[104] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, A Simulator for Confluent P Systems, in [108], 169–184.

[105] A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, Characterizing Tractability with Membrane Creation, in [70], 61–68.

[106] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos Núñez, F.J. Romero-Campero, On the Power of Dissolution in P Systems with Active Membranes, in [86], 373–394.

[107] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero, A Linear Solution for QSAT with Membrane Creation, in [86], 395–409.

[108] M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan (Eds.), *Third Brainstorming Week on Membrane Computing*, Sevilla, GCN TR **01/2005**, University of Seville, 2005.

[109] D. Hauschildt, M. Jantzen, Petri Net Algorithms in the Theory of Matrix Grammars, *Acta Informatica* **31**, 1994, 719–728.

[110] O.H. Ibarra, On Determinism Versus Nondeterminism in P Systems, *Theoretical Computer Science* **344**, 2005, 120–133.

[111] O.H. Ibarra, On Membrane Hierarchy in P Systems, *Theoretical Computer Science* **334**, 2005, 115–129.

[112] O.H. Ibarra, On the Computational Complexity of Membrane Computing Systems, *Theoretical Computer Science* **320**, 1, 2004, 89–109.

[113] O.H. Ibarra, P Systems: Some Recent Results and Research Problems, *Unconventional Programming Paradigms*, International Workshop, UPP 2004, Le Mont Saint Michel, Revised Selected and Invited Papers (J.-P. Banıatre, P. Fradet, J.-L. Giavitto, O. Michel, eds.) *Lecture Notes in Computer Science* **3566**, Springer, 2005, 225–237.

[114] O.H. Ibarra, Some Recent Results Concerning Deterministic P Systems, in [86], 24–25.

[115] O.H. Ibarra, The Number of Membranes Matters, in [27], 273–285.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[116] O.H. Ibarra, Z. Dang, O. Egecioglu, Catalytic Membrane Systems, Semilinear Sets, and Vector Addition Systems, *Theoretical Computer Science* **312**, 2-3, 2004, 378–400.

[117] O.H. Ibarra, Z. Dang, O. Egecioglu, G. Saxena, Characterizations of Catalytic Membrane Computing Systems, *Mathematical Foundations of Computer Science* 2003, 28th International Symposium, MFCS 2003, Bratislava (B. Rovan, P. Vojtás, eds.), *Lecture Notes in Computer Science* **2747**, Springer, 2003, 480–489.

[118] O.H. Ibarra, A. Păun, Counting Time in Computing with Cells, *DNA11* Proceedings, London, Canada, 2005.

[119] O.H. Ibarra, Gh. Păun, Characterizations of Context-Sensitive Languages and Other Language Classes in Terms of Symport/Antiport P Systems, submitted, 2005.

[120] O.H. Ibarra, S.Woodworth, On Bounded Symport / Antiport P Systems, *DNA11* Proceedings, London, Canada, 2005.

[121] O. Ibarra, S. Woodworth, On Symport / Antiport P Systems with One or Two Symbols, in [70], 75–82, and *IEEE Computer Press*, submitted, 2005.

[122] O.H. Ibarra, S. Woodworth, H. Yen, Z. Dang, On Symport / Antiport Systems and Semilinear Sets, in [86], 312–335.

[123] O.H. Ibarra, H.-C. Yen, On Deterministic Catalytic P Systems, *Implementation and Application of Automata*, 10th International Conference, CIAA 2005, Sophia Antipolis, Revised Selected Papers (J. Farré, I. Litovsky, S. Schmitz, eds.), *Lecture Notes in Computer Science* **3845**, Springer, 2006, 163–175.

[124] O.H. Ibarra, H.-C. Yen, Z. Dang, The Power of Maximal Parallelism in P Systems, in [53], 212–224.

[125] M. Ionescu, C. Martín-Vide, Gh. Păun, P Systems with Symport / Antiport Rules: The traces of objects, in [174], and *Grammars* **5**, 2002, 65–79.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[126] M. Ionescu, C. Martín-Vide, A. Păun, Gh. Păun, Membrane Systems with Symport / Antiport. (Unexpected) Universality Results, *DNA Computing*, 8th International Workshop on DNA Based Computers, DNA8, Sapporo, Revised Papers (M. Hagiya, A. Ohuchi, eds.), *Lecture Notes in Computer Science* **2568**, Springer, 2003, 281–290.

[127] M. Ionescu, C. Martín-Vide, A. Păun, Gh. Păun, Unexpected Universality Results for Three Classes of P Systems with Symport / Antiport, *Natural Computing* **2**, 4, 2003, 337–348.

[128] M. Ionescu, D. Sburlan, On P Systems with Promoters/Inhibitors, in [167], 264-280, and *Journal of Universal Computer Science* **10**, 5, 2004, 581–599.

[129] M. Ito, C. Martín-Vide, Gh. Păun, A Characterization of Parikh Sets of ET0L Languages in Terms of P Systems, *Words, Semigroups, and Transducers* (M. Ito, Gh. Paun, S. Yu, eds.), World Scientific, Singapore, 2001, 239–254.

[130] N. Jonoska, N.C. Seeman (Eds.), *DNA Computing*, 7th International Workshop on DNA-Based Computers, DNA7, Tampa, Revised Papers, *Lecture Notes in Computer Science* **2340**, Springer, 2002.

[131] L. Kari, C. Martín-Vide, A. Păun, On the Universality of P Systems with Minimal Symport / Antiport Rules, *Aspects of Molecular Computing* - Essays dedicated to Tom Head on the occasion of his 70th birthday (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), *Lecture Notes in Computer Science* **2950** Festschrift, Springer, 2004, 254–265.

[132] J. Kleijn, M. Koutny, G. Rozenberg, Towards a Petri Net Semantics for Membrane Systems, in [86], 439–460.

[133] S. N. Krishna, *Languages of P Systems. Computability and Complexity*, **PhD Thesis**, Indian Institute of Technology, Madras, India, 2002.

[134] S.N. Krishna, A. Păun, Some Universality Results on Evolution–Communication P Systems, in [63], 207–215.

[135] S.N. Krishna, A. Păun, Three Universality Results on P Systems, in [63], 198–206.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[136] M. Kudlek, V. Mitrana, Some Considerations on a Multiset Model for Membrane Computing, in [174], 311-316, and in [170], 352–359.

[137] M. Madhu, *Studies of P Systems as a Model of Cellular Computing*, **PhD Thesis**, Dept. of Computer Science and Engineering, Indian Institute of Technology, Madras, India, 2003.

[138] M. Margenstern, V. Rogozhin, Y. Rogozhin, S. Verlan, About P Systems with Minimal Symport / Antiport Rules and Four Membranes, *Pre-proceedings of the Fifth Workshop on Membrane Computing*, Milano, 2004, 283–294.

[139] C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.) *Membrane Computing*, International Workshop, WMC 2003, Tarragona, Revised Papers, *Lecture Notes in Computer Science* **2933**, Springer, 2004.

[140] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez-Patón, Tissue P Systems, *Theoretical Computer Science* **296**, 2, 2003, 295–326.

[141] C. Martín-Vide, A. Păun, Gh. Păun, On the Power of P Systems with Symport Rules, *Journal of Universal Computer Science* **8**, 2, 2002, 317–331.

[142] C. Martín-Vide, Gh. Păun, Computing with Membranes (P Systems): Universality Results, Proceedings of 3rd Int'l Conf. *Machines, Computability and Universality*, Chişinău, 2001, *Lecture Notes in Computer Science* **2055** (M. Margenstern, Yu. Rogozhin, eds.), Springer-Verlag, 2001, 82–101.

[143] C. Martín-Vide, Gh. Păun, *Elements of Formal Language Theory for Membrane Computing*, GRLMC Report **21/01**, Rovira i Virgili University, Tarragona, 2001.

[144] C. Martín-Vide, Gh. Păun (Eds.), *Pre-proceedings of Workshop on Membrane Computing*, Curtea de Argeş, 2001, GRLMC Report **16/01**, Rovira i Virgili University, 2001.

[145] G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), *Membrane Computing*. International Workshop, WMC5,

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

Milan, 2004. Revised Papers, *Lecture Notes in Computer Science* **3365**, Springer-Verlag, Berlin, 2005.

[146] G. Mauri, Gh. Păun, C. Zandron (Eds.), Pre-proceedings of the *Fifth Workshop on Membrane Computing* (WMC5), Milano, Universitá di Milano-Bicocca, 2004.

[147] M.L. Minsky, *Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.

[148] I.A. Nepomuceno-Chamorro, A Java Simulator for Basic Transition P Systems, in [167], 309–315, and *Journal of Universal Computer Science* **10**, 5, 2004, 620–619.

[149] M. Oswald, *P Automata*, **PhD Thesis**, Faculty of Computer Science, TU Vienna, 2003.

[150] L. Pan, A. Alhazov, Solving HPP and SAT by P Systems with Active Membranes and Separation Rules, *IEEE Transactions on Computers*, accepted, 2005.

[151] L. Pan, A. Alhazov, T.-O. Ishdorj, Further Remarks on P Systems with Active Membranes, Separation, Merging, and Release Rules, in [167], 316–324, and *Soft Computing - A Fusion of Foundations, Methodologies and Applications* **9**, 9, 2005, 686–690.

[152] C.H. Papadimitrou, *Computational Complexity*, Addison Wesley, 1994.

[153] A. Păun, On P Systems with Global Rules, in [130], 329–339.

[154] A. Păun, On P Systems with Membrane Division, *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 187–201.

[155] A. Păun, *Unconventional Models of Computation: DNA and Membrane Computing*, **PhD Thesis**, Department of Computer Science, The University of Western Ontario, Canada, 2003.

[156] A. Păun, Gh. Păun, The Power of Communication: P Systems with Symport / Antiport, *New Generation Computing* **20**, 2002, 295–305.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[157] A. Păun, Gh. Păun, G. Rozenberg, Computing by Communication in Networks of Membranes, *International Journal of Foundations of Computer Science* **13**, 6, 2002, 779–798.

[158] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, **61**, 1, 2000, 108–143, and Turku Center for Computer Science- *TUCS Report* **208**, 1998.

[159] Gh. Păun, Computing with Membranes: Attacking NP-Complete Problems, *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 94–115.

[160] Gh. Păun, Computing with Membranes (P Systems): Twenty Six Research Topics, *CDMTCS research report* **119**, Univ. of Auckland, 2000.

[161] Gh. Păun, From Cells to Computers: Computing with Membranes (P Systems), Proceedings of Int'l Workshop *Grammar Systems* 2000 (R. Freund, A. Kelemenova, eds.), Bad Ischl, 2000, 9–40, and *BioSystems*, **59**, 3, 2001, 139–158.

[162] Gh. Păun, Further Twenty Six Open Problems in Membrane Computing, in [108], 249–262.

[163] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, Heidelberg, 2002.

[164] Gh. Păun, P Systems with Active Membranes: Attacking NP-Complete Problems, *J. Automata, Languages and Combinatorics*, **6**, 1, 2001, 75–90, and *CDMTCS research report* **102**, Univ. of Auckland, 1999.

[165] Gh. Păun, J. Pazos, M.J. Pérez-Jiménez, A. Rodríguez-Patón, Symport / Antiport P Systems with Three Objects are Universal, *Fundamenta Informaticae* **64**, 2005, 1–4.

[166] Gh. Păun, M.J. Perez-Jiménez, A. Riscos-Núñez, Tissue P Systems with Cell Division, in [167], 380–386.

[167] Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini (Eds.), *Second Brainstorming Week on Membrane Computing*, Sevilla, GCN TR **01/2004**, University of Seville, 2004.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[168] Gh. Păun, G. Rozenberg, A Guide to Membrane Computing, *Theoretical Computer Science* **287**, 1, 2002, 73–100.

[169] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer, Berlin, Heidelberg, 1998.

[170] Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), *Membrane Computing*. International Workshop WMC-CdeA 2002, Curtea de Argeş, Romania, Revised Papers, *Lecture Notes in Computer Science* **2597**, Springer-Verlag, Berlin, 2003.

[171] Gh. Păun, Y. Sakakibara, T. Yokomori, P Systems on Graphs of Restricted Forms, *Publicationes Mathematicae* **60**, 2002.

[172] Gh. Păun, T. Yokomori, Membrane Computing Based on Splicing, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **54**, American Mathematical Society, 1999, 217–232.

[173] Gh. Păun, S. Yu, On Synchronization in P Systems, *Fundamenta Informaticae* **38**, 4, 1999, 397–410, and University of Western Ontario Report **539**, 1999.

[174] Gh. Păun, C. Zandron (Eds.), Pre-proceedings of *Workshop on Membrane Computing*, Curtea de Argeş, MolCoNet Publication No **1**, 2002.

[175] M.J. Pérez-Jiménez, Complexity Classes in Membrane Computing, in [146], 63–63.

[176] M.J. Pérez Jiménez, A. Romero Jiménez, F. Sancho Caparrini, Complexity Classes in Models of Cellular Computing with Membranes, *Natural Computing* **2**, 3, 2003, 265–285.

[177] M.J. Pérez-Jiménez, A. Riscos-Núñez (Eds.), *Modelos de Computacion Molecular, Celular y Cuantica*, Fénix Editorial, Sevilla, 2004.

[178] M.J. Pérez-Jiménez, F.J. Romero-Campero, Trading Polarizations for Bi-stable Catalysts in P Systems with Active Membranes, in [146], 327–342.

[179] M.J. Pérez-Jiménez, A. Romero-Jiménez, Simulating Turing Machines by P Systems, *Fundamenta Informaticae* **49**, 1-3, 2002, 273–287.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[180] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity Classes in Cellular Computing with Membranes, in [63], 270-278, and *Natural Computing* **2**, 3, 2003, 265–285.

[181] M.J. Pérez-Jiménez, F. Sancho-Caparrini, A Formalization of Basic P Systems, *Fundamenta Informaticae* **49**, 1-3, 2002, 261–272.

[182] M. Pérez Jiménez, F. Sancho Caparrini, *Computacion celular con membranas: Un modelo no convencional*, Kronos Editorial, Sevilla, 2002.

[183] I. Petre, A Normal Form for P Systems, *Bulletin of the EATCS* **67**, 1999, 165–172.

[184] Z. Qi, C. Fu, D. Shi, J.You, Specification and Execution of P Systems with Symport / Antiport Rules Using Rewriting Logic, in [146], 363–371.

[185] Z. Qi, J. You, P Systems and Petri Nets, in [27], 387–403.

[186] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[187] M.H. Saier, jr., A Functional-Phylogenetic Classification System for Transmembrane Solute Transporters, *Microbiology and Molecular Biology Reviews*, 2000, 354–411.

[188] A. Salomaa, G. Rozenberg (Eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.

[189] D. Sburlan, A Static Sorting Algorithm for P Systems with Mobile Catalysts, *Analele Stiintifice Univ. Ovidius Constanţa*, seria Matematica, **11**, 1, Constanţa, 2003, 195–205.

[190] D. Sburlan, Clock-free P Systems, in [146], 372–383.

[191] D. Sburlan, Further Results on P Systems with Promoters/Inhibitors, in [108], 289–304.

[192] D. Sburlan, Membrane Systems with Promoters/Inhibitors. From Computational Universality to Algorithms, RNGC Report **04/2004**, Sevilla University, 2004.

UNIVERSITAT ROVIRA I VIRGILI
COMMUNICATION IN MEMBRANA SYSTEMS WITH SYMBOL OBJECTS.
Artiom Alhazov
ISBN: 978-84-690-7630-9 / DL: T.1400-2007

[193] D. Sburlan, Non-cooperative P Systems with Priorities Characterize PsET0L, in [86], 530–539.

[194] D. Sburlan, *Promoting and Inhibiting Contexts in Membrane Computing*, **PhD Thesis**, University of Sevilla, Spain, 2006.

[195] P. Sosík, The Power of Catalysts and Priorities in Membrane Systems, *Grammars* **6**, 1, 2003, 13–24.

[196] P. Sosík, R. Freund, P Systems Without Priorities are Computationally Universal, in [170], 400–409.

[197] P. Sosík, Solving a PSPACE-Complete Problem by P Systems with Active Membranes, in [63], 305–312.

[198] P. Sosík, R. Freund, P Systems Without Priorities are Computationally Universal, in [170], 400–409.

[199] A. Syropoulos, E.G. Mamatas, P.C. Allilomes, K.T. Sotiriades, A Distributed Simulation of P Systems, in [27], 455–460.

[200] Gy. Vaszil, On the Size of P Systems with Minimal Symport / Antiport, in [146], 422–431, and in [145], 404–413.

[201] S. Verlan, *Head Systems and Application to Bio-informatics*, **PhD Thesis**, LITA, Univ. Metz, 2004.

[202] S. Verlan, Optimal Results on Tissue P Systems with Minimal Symport / Antiport, Presented at *EMCC meeting*, Lorentz Center, Leiden, 2004.

[203] S. Verlan, Tissue P Systems with Minimal Symport / Antiport, in [53], 418–430.

[204] C. Zandron, *A Model for Molecular Computing: Membrane Systems*, **PhD Thesis**, Dipartimento di Scienze dell'Informazione, Universitá degli Studi di Milano, Italy, 2002.

[205] P Systems Web Page, `http://psystems.disco.unimib.it`