Typed Membrane Systems

Bogdan Aman and Gabriel Ciobanu

[1] Romanian Academy, Institute of Computer Science
[2] A.I.Cuza University of Iaşi, Romania
`baman@iit.tuiasi.ro, gabriel@info.uaic.ro`

Summary. We introduce and study typing rules and a type inference algorithm for P systems with symport/antiport evolution rules. The main results are given by a subject reduction theorem and the completeness of type inference. We exemplify how the type system is working by presenting a typed description of the sodium-potassium pump.

1 Introduction

Membrane systems (also called P systems) were introduced by Gh. Păun; several variants of P systems are presented in the monograph [8]. P systems are parallel and nondeterministic computing models inspired by the compartments of eukaryotic cells and by their biochemical reactions. The structure of the cell is represented by a set of hierarchically embedded regions, each one delimited by a surrounding boundary (called membrane), and all of them contained inside an external special membrane called *skin*. The molecular species (ions, proteins, etc.) floating inside cellular compartments are represented by multisets of objects described by means of symbols or strings over a given alphabet, objects which can be modified or communicated between adjacent compartments. Chemical reactions are represented by evolution rules which operate on the objects, as well as on the compartmentalized structure (by dissolving, dividing, creating, or moving membranes).

A P system can perform computations in the following way: starting from an initial configuration which is defined by the multiset of objects initially placed inside the membranes, the system evolves by applying the evolution rules of each membrane in a nondeterministic and maximally parallel manner. A rule is applicable when all the objects that appear in its left hand side are available in the region where the rule is placed. The maximally parallel way of using the rules means that in each step, in each region of the system, we apply a maximal multiset of rules, namely a multiset of rules such that no further rule can be added to the set. A halting configuration is reached when no rule is applicable. The result is represented by the number of objects from a specified membrane.

Several variants of P systems are inspired by different aspects of living cells (symport and antiport-based communication through membranes, catalytic objects, membrane charge, etc.). Their computing power and efficiency have been investigated using the approaches of formal languages and grammars, register machines and complexity theory. An updated bibliography can be found at the P systems web page [10].

P systems are known to be Turing complete [8]. They are also used to model biological systems and their evolution [5]. A type description of calculus of looping sequences, along with a type inference algorithm can be found in [2]. Related static techniques have been applied to biological systems, such as Control Flow Analysis [7] and Abstract Interpretation [6]. In this paper we define a typing system and a type inference algorithm for P systems with symport/antiport evolution rules. To exemplify how the introduced type system works, we use types in the description of the sodium-potassium pump.

The cells of the human body have different *types* depending on the morphological or functional form [1]. A complete list of distinct cell types in the adult human body may include about 210 distinct types. The chemical reactions inside cells are usually expressed by using types of the components; for instance, a reaction between an *acid* and a *carbonate* forms *salt*, *carbon dioxide* and *water* as the only products. In this paper we enrich the symport/antiport P systems with a *type discipline*. The key technical tools are type inference and principal typing [9]; we associate to each reduction rule a minimal set of conditions that must be satisfied in order to assure that applying this rule to a correct P system, we get a correct membrane system as well. The type system for P systems with symport/antiport rules is (up to our knowledge) the first attempt to control the evolution of P systems using typing rules. The presentation of the typed sodium-potassium pump is an example how to introduce and use types in P systems.

The structure of the paper is as follows. A type system for membranes with symport/antiport rules is introduced in Section 2. Section 3 contains an extension of the description of the sodium-potassium pump using P systems [3] with the newly introduced type system. The section ends with an example of a rule that would be considered ill-typed for the pump. Conclusion and references end the paper.

2 Typed Discipline for Membrane Systems

A type system is used to prevent the occurrences of errors during the evolution of a system. A type inference procedure determines the minimal requirements to accept a system or a component as well-typed. These are important concepts and methods of programming languages and software engineering. In this paper, we investigate the application of these concepts to a biologically inspired formalism, namely to membrane systems.

We use membrane systems with symport/antiport rules. From biological observations we know that there are many cases where two chemicals pass through a membrane at the same time, with the help of each other, either in the same direction, or in opposite directions; in the former case we say that we have a *symport*, in the latter case we have an *antiport*. Symport is standardly described by rules of the form (ab, in) and (ab, out) associated with a membrane, that state that the objects a and b can enter, respectively, exit the membrane together; antiport is described by rules of form $(a, out; b, in)$ associated with a membrane, that state that a exits at the same time when b enters the membrane. Inspired by the rules for active membranes [8], and the notation used in [3], we denote the symport rules by $ab[_l \rightarrow [_l ab$ or $[_l ab \rightarrow ab[_l$, and the antiport rules by $b[_l a \rightarrow a[_l b$. Generalizing such kinds of rules, we can consider rules of the unrestricted forms $x[_l \rightarrow [_l x$ or $[_l y \rightarrow y[_l$ (generalized symport rules), and $x[_l y \rightarrow y[_l x$ (generalized antiport rules), where x, y are strings representing multisets of objects (without any restriction on the length), and l is the label of the membrane in which the rules are placed. It is worth to note that an antiport rule with one of x, y empty is just a symport rule.

Definition 1. *A* membrane system with symport/antiport rules *is a construct*
$$\Pi = (O, H, \mu, w_1, \ldots, w_n, E, R_1, \ldots, R_n, i_O)$$
where:

- $n \geq 1$ *(the initial* degree *of the system);*
- O *is an alphabet (its elements are called* objects*);*
- H *is a finite set of* labels *for membranes;*
- $\mu \subset H \times H$ *describes the* membrane structure*, such that* $(i, j) \in \mu$ *denotes that the membrane labelled by* j *is contained in the membrane labelled by* i*; we distinguish the external membrane (usually called the "skin" membrane) and several internal membranes;*
- $w_1, \ldots, w_n$ *are strings over* O*, describing the* multisets of objects *placed in the* n *regions of* μ*;*
- $E \subseteq O$ *is the set of objects which are supposed to appear in the environment in arbitrarily many copies;*
- R_i*,* $1 \leq i \leq n$ *is a finite set of symport and antiport rules over* O *associated with the* n *membranes of* μ*;*
- i_O*,* $1 \leq i_O \leq n$ *is the output membrane.*

Definition 2. *The set* $\mathcal{M}(\Pi)$ *of membranes in a P system* Π *is inductively defined as follows:*

- *if* i *is a label from* H *and* w *is a multiset over* O *then* $[w]_i \in \mathcal{M}(\Pi)$*;* $[w]_i$ *is called an* elementary membrane*;*
- *if* i *is a label from* H*,* $M_1, \ldots, M_n \in \mathcal{M}(\Pi)$*,* $n \geq 1$*, and* w *is a multiset over* O *then* $[w\ M_1 \ldots M_n]_i \in \mathcal{M}(\Pi)$*;* $[w\ M_1 \ldots M_n]_i$ *is called a* composite membrane.

Definition 3. *For a P system* Π*, if* M *and* N *are two membranes from* $\mathcal{M}(\Pi)$*, we say that* M *reduces to* N *(*$M \rightarrow N$*) if there exists a rule in a* R_i*,* $1 \leq i \leq n$*, applicable to membrane* M *such that we can obtain membrane* N*.*

More details on membrane systems can be found in [8].

2.1 Typed Membrane Systems

We introduce typing rules for the class of membrane systems with symport/ antiport rules in Table 1 and Table 2. We use *obj* to denote objects, u and v to denote multisets of objects, and *mem* to denote membranes. The main judgements normally take the form

$$\Gamma \vdash M : T$$

indicating that a membrane denoted by M is a well-typed system having the type T relative to a typing environment Γ.

The steps for defining a type system are as follows:

1. For each object *obj* we establish a certain type T.
2. A membrane *mem* has a type $\{S, D^{\uparrow}, D^{\downarrow}, L\}$, where:
   - S is a set of object types representing the objects that are allowed to stay in membrane *mem* during all the possible evolutions of the system;
   - $D^{\uparrow}$ is a set of sets of object types representing the objects that are allowed to be communicated up through membrane *mem* during all the possible evolutions of the system;
   - $D^{\downarrow}$ is a set of sets of object types representing the objects that are allowed to be communicated down through membrane *mem* during all the possible evolutions of the system;
   - L is a set of labels denoting certain states of the membrane *mem* during all the possible evolutions of the systems.

These steps are exemplified in Section 3 for a sodium-potassium pump.

Table 1: *Typing Rules for Membrane Systems*

$$\frac{obj : T \in \Gamma}{\Gamma \vdash obj : T}\,\textbf{(R1)}$$

$$\frac{\begin{array}{c} mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\} \in \Gamma \quad [u_1 \dots u_i \;\; mem_1 \dots mem_j]^{l}_{mem} \\ \Gamma \vdash u_1 : T_1 \dots \Gamma \vdash u_i : T_i \quad \{T_1, \dots, T_i\} \subseteq S_{mem} \quad l \in L_{mem} \\ \Gamma \vdash mem_1 : \{S_1, D^{\uparrow}_1, D^{\downarrow}_1, L_1\} \dots \Gamma \vdash mem_j : \{S_j, D^{\uparrow}_j, D^{\downarrow}_j, L_j\} \end{array}}{\Gamma \vdash mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\}}\,\textbf{(R2)}$$

Rule **(R1)** states that an object is a well behaved object if it is typed in Γ. Rule **(R2)** states that if we have some membranes (possibly none) and some objects u_i which together are well behaved in an environment Γ, and $u_1 \dots u_i$ can stay in a membrane *mem* which is also well formed in the environment Γ, and a label l can be associated to the membrane *mem*, then also $[u_1 \dots u_i \;\; mem_1 \dots mem_j]^{l}_{mem}$ is well formed under the assumptions of Γ.

Lemma 1 (Generation Lemma).

1. *If $\Gamma \vdash obj : T$, then $obj : T \in \Gamma$.*

2. If $\Gamma \vdash mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\}$, then we have that mem is a membrane $[u_1 \dots u_i \quad mem_1 \dots mem_j]^l_{mem}$ with $\Gamma \vdash u_1$: $T_1 \dots \Gamma \vdash u_i : T_i$ $\Gamma \vdash mem_1 : \{S_1, D^{\uparrow}_1, D^{\downarrow}_1, L_1\} \dots \Gamma \vdash mem_j : \{S_j, D^{\uparrow}_j, D^{\downarrow}_j, L_j\}, \{T_1, \dots, T_i\} \subseteq S_{mem}, l \in L_{mem}$, $mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\} \in \Gamma$.

Proof. By induction on the depth of the membranes.

In Table 2 we describe the type conditions the rules from the class of membrane systems with symport/antiport rules must fulfill such that the evolution takes place as expected. Some notations are necessary: $mem1$ is the parent membrane of $mem2$, and by $T_u = \{T_{u_1}, \dots, T_{u_i}\}$ we denote the set of types of a multiset of objects $u = u_1, \dots, u_i$. In these rules, l' can be the same l (meaning that the state of the membrane does not change).

Table 2: *Typed Evolution Rules for Membrane Systems*

$$\frac{\begin{array}{c} u[^l_{mem2} \to [^{l'}_{mem2} u \quad \Gamma \vdash u : T_u \quad T_u \subseteq D^{\downarrow}_{mem2} \\ T_u \subseteq S_{mem1} \quad T_u \subseteq S_{mem2} \quad l' \in L_{mem2} \\ mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \in \Gamma \\ mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \in \Gamma \end{array}}{\begin{array}{c} \Gamma \vdash mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \\ \Gamma \vdash mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \end{array}} \textbf{(R3)}$$

$$\frac{\begin{array}{c} [^l_{mem2} u \to u[^{l'}_{mem2} \quad \Gamma \vdash u : T_u \quad T_u \subseteq D^{\uparrow}_{mem2} \\ T_u \subseteq S_{mem1} \quad T_u \subseteq S_{mem2} \quad l' \in L_{mem2} \\ mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \in \Gamma \\ mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \in \Gamma \end{array}}{\begin{array}{c} \Gamma \vdash mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \\ \Gamma \vdash mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \end{array}} \textbf{(R4)}$$

$$\frac{\begin{array}{c} v[^l_{mem2} u \to u[^{l'}_{mem2} v \quad \Gamma \vdash u : T_u \quad \Gamma \vdash v : T_v \quad T_u \subseteq D^{\uparrow}_{mem2} \quad T_v \subseteq D^{\downarrow}_{mem2} \\ T_u \subseteq S_{mem1} \quad T_u \subseteq S_{mem2} \quad T_v \subseteq S_{mem1} \quad T_v \subseteq S_{mem2} \quad l' \in L_{mem2} \\ mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \in \Gamma \\ mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \in \Gamma \end{array}}{\begin{array}{c} \Gamma \vdash mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \\ \Gamma \vdash mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \end{array}} \textbf{(R5)}$$

Denoting by M and N two membrane systems, we have the following result:

Theorem 1 (Subject Reduction).
If all the objects and membranes of M are well typed in an environment Γ, and $M \to N$ by applying a rule of Table 2, then N is a membrane system such that all its objects and membranes are well typed in the environment Γ.

Proof (Sketch). Case $[^l_{depth2} u \to u[^{l'}_{depth2}$. We consider $depth1$ to be the parent membrane of $depth2$. If we apply this rule the only structure that changes is

$[[u\ldots]^{l}_{depth2}\ldots]_{depth1}$ which is transformed into $[[\ldots]^{l'}_{depth2}u\ldots]_{depth1}$. If $depth1$: $\{S_{depth1},\ D^{\uparrow}_{depth1},\ D^{\downarrow}_{depth1},\ L_{depth1}\}$ then by Lemma 1 applied twice we have $depth1 : \{S_{depth1},\ D^{\uparrow}_{depth1},\ D^{\downarrow}_{depth1},\ L_{depth1}\} \in \Gamma$, $depth2 : \{S_{depth2},\ D^{\uparrow}_{depth2},\ D^{\downarrow}_{depth2},\ L_{depth2}\} \in \Gamma$, $\Gamma \vdash u : T_u$. Since the rule can be applied, then we get $T_u \subseteq S_{depth2}$, $l' \in L_{depth2}$ and $T_u \subseteq S_{depth1}$. By applying (**R4**) we have that $\Gamma \vdash depth1 : \{S_{depth1},\ D^{\uparrow}_{depth1},\ D^{\downarrow}_{depth1},\ L_{depth1}\}$ and $\Gamma \vdash depth2 : \{S_{depth2},\ D^{\uparrow}_{depth2},\ D^{\downarrow}_{depth2},\ L_{depth2}\}$ which means that all the objects and membranes of N are well typed in the environment Γ.

The other cases are treated similarly.

2.2 Type Inference Algorithm

Given a raw membrane system M, i.e., a well-formed membrane system in which all type annotations have been erased, our type inference algorithm introduces the needed type annotations and computes the environment satisfying the minimal requirements on the typing of the objects and membranes occurring in M, thus producing M' which is well typed with respect to such environment. The typing given by M' is principal in the sense of [9], since all other possible typings which can be given to membrane systems obtained from M by introducing type annotations can be derived through a set of suitable operations from the inferred typing of M'. The inference algorithm is then proved to be sound and complete with respect to the rules of Subsection 2.1.

Types and type environments of the algorithm are related to the structure of the system; it has therefore to put together distinct environments whenever the system has more than one parallel structure.

The type reconstruction procedure is represented by a judgement

$$\vdash_I M : \langle W, \Gamma \rangle,$$

where M is a membrane structure, W is the type inferred for M from the environment Γ, and I represents the fact that this judgement results from the inference algorithm. As before, we consider that $mem1$ is the parent membrane of $mem2$. We define the domain of a set of typed names Γ as

$$dom(\Gamma) = \{n \mid n : t \in \Gamma\}.$$

where t is the name of an object or membrane.

We say that two typed sets of names Γ and Γ' are *compatible* (written $\Gamma \bowtie \Gamma'$) if and only if $n : t \in \Gamma$ and $n : t' \in \Gamma'$, then it holds $t = t'$. The disjoint union of Γ and Γ' is defined as

$$\Gamma \uplus \Gamma' = \{n : t \in \Gamma \wedge n \notin dom(\Gamma')\} \cup \{n : t' \in \Gamma' \wedge n \notin dom(\Gamma)\}.$$

We also define a function that returns the type of an object or a membrane with respect to a type environment Γ:

$$type(n, \Gamma) = \{n : t \mid n : t \in \Gamma\}$$

The inference procedure is defined in a natural semantic style. In all the type inference rules, the objects and membrane types which appear in conclusions are derived from those appearing in premises.

Table 3: *Type reconstruction*

$$\vdash_I obj : \langle Obj, obj : Obj\rangle \;\textbf{(I1)}$$

$$\frac{\begin{array}{c}[u_1 \dots u_i\; mem_1 \dots mem_j]^l_{mem} \quad \Gamma_s \bowtie \Gamma_t, s \neq t, 1 \leq s, t \leq i+j \\ \vdash_I u_1 : \langle T_{u_1}, \Gamma_1\rangle \dots \vdash_I u_i : \langle T_{u_i}, \Gamma_i\rangle \\ \vdash_I mem : \langle T, \Gamma\rangle \quad \vdash_I mem_1 : \langle T_1, \Gamma_{i+1}\rangle \dots \vdash_I mem_j : \langle T_j, \Gamma_{i+j}\rangle\end{array}}{\begin{array}{c}\vdash_I mem : \langle T', \Gamma'\rangle \\ \text{where } T' = \{S_{mem} \cup \{T_{u_1}, \dots, T_{u_i}\}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem} \cup \{l\}\} \\ \text{if } T = \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\} \\ \text{and } \Gamma' = \Gamma \cup (\biguplus_{k=1}^{i+j} \Gamma_k \backslash type(mem, \biguplus_{k=1}^{i+j} \Gamma_k)) \cup \{mem : T'\}\end{array}} \;\textbf{(I2)}$$

$$\frac{\begin{array}{c}u[^{l}_{mem2}\!\rightarrow [^{l'}_{mem2}u \quad \Gamma \bowtie \Gamma_2 \quad \Gamma \bowtie \Gamma_1 \\ \vdash_I u : \langle T_u, \Gamma\rangle \quad \vdash_I mem1 : \langle\{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma_1\rangle \\ \vdash_I mem2 : \langle\{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\}, \Gamma_2\rangle\end{array}}{\begin{array}{c}\vdash_I mem1 : \langle\{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma'_1\rangle \\ \text{where } \Gamma'_1 = ((\Gamma_1 \uplus \Gamma)\backslash type(mem1, \Gamma \uplus \Gamma_1)) \cup \\ \{mem1 : \{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}\} \\ \vdash_I mem2 : \langle\{S_{mem2} \cup T_u, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2} \cup T_u, L_{mem2} \cup \{l'\}\}, \Gamma'_2\rangle \\ \text{where } \Gamma'_2 = ((\Gamma_2 \uplus \Gamma)\backslash type(mem2, \Gamma_2 \uplus \Gamma)) \cup \\ \{mem2 : \{S_{mem2} \cup T_u, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2} \cup T_u, L_{mem2} \cup \{l'\}\}\}\end{array}} \;\textbf{(I3)}$$

$$\frac{\begin{array}{c}[^{l}_{mem2}u \rightarrow u[^{l'}_{mem2} \quad \Gamma_1 \bowtie \Gamma_2 \quad \Gamma \bowtie \Gamma_1 \quad \Gamma \bowtie \Gamma_2\} \\ \vdash_I u : \langle T_u, \Gamma\rangle; \emptyset \quad \vdash_I mem1 : \langle\{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma_1\rangle \\ \vdash_I mem2 : \langle\{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\}, \Gamma_2\rangle\end{array}}{\begin{array}{c}\vdash_I mem1 : \langle\{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma'_1\rangle \\ \text{where } \Gamma'_1 = ((\Gamma \uplus \Gamma_1 \uplus \Gamma_2)\backslash type(mem1, \Gamma \uplus \Gamma_1 \uplus \Gamma_2)) \cup \\ \{mem1 : \{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}\} \\ \vdash_I mem2 : \langle\{S_{mem2} \cup T_u, D^{\uparrow}_{mem2} \cup T_u, D^{\downarrow}_{mem2}, L_{mem2} \cup \{l'\}\}, \Gamma'_2\rangle \\ \text{where } \Gamma'_2 = ((\Gamma \uplus \Gamma_1 \uplus \Gamma_2)\backslash type(mem2, \Gamma \uplus \Gamma_1 \uplus \Gamma_2)) \cup \\ \{mem2 : \{S_{mem2} \cup T_u, D^{\uparrow}_{mem2} \cup T_u, D^{\downarrow}_{mem2}, L_{mem2} \cup \{l'\}\}\}\end{array}} \;\textbf{(I4)}$$

$$\frac{\begin{array}{c}v[^{l}_{mem2}u \rightarrow u[^{l'}_{mem2}v \quad \Gamma_i \bowtie \Gamma_j, i \neq j \quad \vdash_I u : \langle T_u, \Gamma_3\rangle \quad \vdash_I v : \langle T_v, \Gamma_4\rangle \\ \vdash_I mem1 : \langle\{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma_1\rangle \\ \vdash_I mem2 : \langle\{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\}, \Gamma_2\rangle\end{array}}{\begin{array}{c}\vdash_I mem1 : \langle\{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}, \Gamma'_1\rangle \\ \text{where } \Gamma'_1 = ((\biguplus_{i=1;i\neq 2}^{4} \Gamma_i)\backslash type(mem1, \biguplus_{i=1;i\neq 2}^{4} \Gamma_i)) \cup \\ \{mem1 : \{S_{mem1} \cup T_u, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}\} \\ \vdash_I mem2 : \langle\{S_{mem2} \cup T_v, D^{\uparrow}_{mem2} \cup T_u, D^{\downarrow}_{mem2} \cup T_v, L_{mem2} \cup \{l'\}\}, \Gamma'_2 \\ \text{where}\Gamma'_2 = ((\biguplus_{i=2}^{4} \Gamma_i)\backslash type(mem2, \biguplus_{i=2}^{4} \Gamma_i)) \cup \\ \{mem2 : \{S_{mem2} \cup T_v, D^{\uparrow}_{mem2} \cup T_u, D^{\downarrow}_{mem2} \cup T_v, L_{mem2} \cup \{l'\}\}\}\end{array}} \;\textbf{(I5)}$$

Using rules of the form (**I1**) to each object obj of a given membrane system, we attach a fresh type Obj. If we add two different types Obj_1 and Obj_2 to the same object obj when constructing the type of the whole membrane system using rules of Table 3, by using the relation $\bowtie$ we get $Obj_1 = Obj_2$. Rules (**I3**), (**I4**) and (**I5**) are used to construct the types of the membranes with conditions given by symport and antiport rules that can be applied, while rule (**I2**) is used to update the type of membranes.

A subtyping relation $\leq$ is introduced to compare the environments. If we take two type environments $\Gamma = \{a : K, b : Na\}$ and $\Delta = \{a : K\}$, then $\Gamma \leq \Delta$.

Theorem 2 (Soundness of the Type Inference).
If $\vdash_I M : \langle W, \Gamma \rangle$, *then* $\Gamma \vdash M : W$.

Proof. By induction on the structure of deductions in $\vdash_I$.

- Case (**I1**): We have $\vdash_I obj : \langle Obj, obj : Obj \rangle$, from where it results that $obj : Obj \in \Gamma$. Applying rule (**R1**) it results that $\Gamma \vdash obj : Obj$.
- Case (**I2**): We have
  (i) the membrane structure $[u_1 \dots u_i mem_1 \dots mem_j]^l_{mem}$;
  (ii) from $\vdash_I u_1 : \langle T_{u_1}, \Gamma_1 \rangle \dots \vdash_I u_i : \langle T_{u_i}, \Gamma_i \rangle$; $\Gamma_k \bowtie \Gamma_t, 1 \leq k, t \leq i$; $\Gamma \leq \Gamma_k$, $1 \leq k \leq i$ applying the induction we have that $\Gamma \vdash u_1 : T_{u_1} \dots \Gamma \vdash u_i : T_{u_i}$;
  (iii) from $\vdash_I mem : \langle T, \Gamma_i \rangle \quad \vdash_I mem_1 : \langle T_1, \Gamma_{i+1} \rangle \dots \vdash_I mem_j : \langle T_j, \Gamma_{i+j} \rangle$; $\Gamma_k \bowtie \Gamma_t, i+1 \leq k, t \leq i+j$; $\Gamma \leq \Gamma_k$, $i+1 \leq k \leq i+j$ applying the induction we have that $\Gamma \vdash mem_1 : \{S_1, D_1^{\uparrow}, D_1^{\downarrow}, L_1\} \dots \Gamma \vdash mem_j : \{S_j, D_j^{\uparrow}, D_j^{\downarrow}, L_j\}$;
  (iv) $mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\} \in \Gamma$;
  (v) $\{T_{u_1}, \dots, T_{u_i}\} \subseteq S_{mem}$, $l \in L_{mem}$.
  Using (i), (ii), (iii), (iv), and (v), we can apply rule (**R2**), and so obtaining $\Gamma \vdash mem : \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\}$.
- Case (**I3**): For membrane $mem2$ we have
  (i) the rule $u[^l_{mem2} \rightarrow [^{l'}_{mem2} u$;
  (ii) from $\vdash_I u : \langle T_u, \Gamma \rangle$, $\Gamma \bowtie \Gamma'_2$, $\Gamma'_2 \leq \Gamma$ applying the induction we have that $\Gamma'_2 \vdash u : T_u$;
  (iii) $mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\} \in \Gamma'_2$;
  (iv) $T_u \subseteq D^{\downarrow}_{mem2}$, $T_u \subseteq S_{mem2}$, $l' \in L_{mem2}$.
  Using (i), (ii), (iii) and (iv), we can apply rule (**R3**) and obtain $\Gamma \vdash mem2 : \{S_{mem2}, D^{\uparrow}_{mem2}, D^{\downarrow}_{mem2}, L_{mem2}\}$.
  For membrane $mem1$ we have
  (i) the rule $u[^l_{mem2} \rightarrow [^{l'}_{mem2} u$;
  (ii) from $\vdash_I u : \langle T_u, \Gamma \rangle$, $\Gamma \bowtie \Gamma'_1$, $\Gamma'_1 \leq \Gamma$ applying the induction we have that $\Gamma'_1 \vdash u : T_u$;
  (iii) $mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\} \in \Gamma'_1$;
  (iv) $T_u \subseteq S_{mem1}$.
  Using (i), (ii), (iii) and (iv), we can apply rule (**R3**) and obtain $\Gamma \vdash mem1 : \{S_{mem1}, D^{\uparrow}_{mem1}, D^{\downarrow}_{mem1}, L_{mem1}\}$.

The other cases are treated in a similar manner.

Theorem 3 (Completeness of the Type Inference).
If $\Gamma \vdash M : W$, then $\vdash_I M : \langle W', \Gamma' \rangle$, and there is a a renaming function σ such that:

1. *$\sigma(W') = W$;*
2. *$\sigma(\Gamma') \leq \Gamma$.*

Proof. By induction on the structure of deductions in $\vdash$.

- Case (**R1**): From (**R1**) we have that $\Gamma \vdash obj : Obj$, while from (**I1**) we have that $\vdash_I obj : \langle Obj', obj : Obj' \rangle$. If we consider $\sigma(Obj') = Obj$, $\sigma(obj : Obj') = obj : Obj$ we get that $\vdash_I obj : \langle Obj, obj : Obj \rangle$.
- Case (**R2**): We have
  (i) the membrane structure $[u_1 \dots u_i mem_1 \dots mem_j]^l_{mem}$;
  (ii) from $\Gamma \vdash u_1 : T_{u_1} \dots \Gamma \vdash u_i : T_{u_i}$ applying the induction we have that $\vdash_I u_1 : \langle T_{u_1}, \Gamma_1 \rangle \dots \vdash_I u_i : \langle T_{u_i}, \Gamma_i \rangle$; $\sigma(\Gamma_k) \leq \Gamma$, $1 \leq k \leq i$;
  (iii) from $\Gamma \vdash mem_1 : \{S_1, D_1^{\uparrow}, D_1^{\downarrow}, L_1\} \dots \Gamma \vdash mem_j : \{S_j, D_j^{\uparrow}, D_j^{\downarrow}, L_j\}$ applying the induction we have that $\vdash_I mem : \langle T, \Gamma_i \rangle \quad \vdash_I mem_1 : \langle T_1, \Gamma_{i+1} \rangle \dots \vdash_I mem_j : \langle T_j, \Gamma_{i+j} \rangle$; $\sigma(\Gamma_k) \leq \Gamma$, $i+1 \leq k \leq i+j$;
  (iv) $mem : T \in \Gamma$, where $T = \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\}$;
  (v) $\{T_{u_1}, \dots, T_{u_i}\} \subseteq S_{mem}$, $l \in L_{mem}$.
  Using (i), (ii), (iii), (iv) and (v), we can apply rule (**I2**) and obtain $\vdash_I mem : \langle T', \Gamma' \rangle$, where $T' = \{S_{mem}, D^{\uparrow}_{mem}, D^{\downarrow}_{mem}, L_{mem}\}$ and $\Gamma' = \Gamma \cup \biguplus_{k=1}^{i+j} \Gamma_k$. We have that $T' = T$ and $\sigma(\Gamma') \leq \Gamma$.

The other cases are treated in a similar manner.

3 Na-K Pump Modelled by Typed Membranes

The sodium-potassium pump is a primary active transport system driven by a cell membrane ATPase carrying sodium ions out and potassium ions in. The description given in Table 4; it is known as the Albers-Post model. According to this mechanism:

1. Na^+ and K^+ transport is similar to a ping-pong mechanism, meaning that the two ions species are transported sequentially;
2. Na-K pump essentially exists in two conformations, E1 and E2, which may be phosphorylated or dephosphorylated.

These conformations correspond to two mutually exclusive states in which the pump exposes ion binding sites alternatively on the cytoplasmic (E1) and extracellular (E2) sides of the membrane. Ion transport is mediated by transitions between these conformations. In Table 4 we use the following notations:

- $A + B$ means that A and B are present together and could react;
- $A \cdot B$ means that A and B are bound to each other non-covalently;

- $E_2 \sim P$ indicates that the phosphoryl group P is covalently bound to E_2;
- P_i is the inorganic phosphate group;
- $\rightleftharpoons$ indicates that the process can also proceed in a reversible way.

Table 4: *The Albers-Post Model*

$$
\begin{array}{rcll}
E_1 + Na^+_{in} & \rightleftharpoons & Na^+ \cdot E_1 & (1)\\
Na^+ \cdot E_1 + ATP & \rightleftharpoons & Na^+ \cdot E_1 \sim P + ADP & (2)\\
Na^+ \cdot E_1 \sim P & \rightleftharpoons & Na^+ \cdot E_2 \sim P & (3)\\
Na^+ \cdot E_2 \sim P & \rightleftharpoons & E_2 \sim P + Na^+_{out} & (4)\\
E_2 \sim P + K^+_{in} & \rightleftharpoons & K^+ \cdot E_2 \sim P & (5)\\
K^+ \cdot E_2 \sim P & \rightleftharpoons & K^+ \cdot E_2 + P_i & (6)\\
K^+ \cdot E_2 & \rightleftharpoons & K^+ \cdot E_1 & (7)\\
K^+ \cdot E_1 & \rightleftharpoons & K^+_{in} + E_1 & (8)
\end{array}
$$

3.1 The Membrane Systems Model of the Pump

The environment and the inner region are characterized by multisets of symbols over the alphabet $V = \{Na, K, ATP, ADP, P\}$, representing the substances floating inside them. The conformations of the pump are described by means of labels attached to the membrane, that is $[\,]_l$ with $l \in L$, $L = \{E_1, E_2, E_1^P, E_2^P\}$. The labels E_1, E_2 correspond to the dephosphorylated conformations of the pump, while E_1^P, E_2^P correspond to the phosphorylated conformations. Note an important aspect of this system: the object P now becomes part of the membrane label, hence it undergoes a structural modification by passing from being an element of the alphabet V to being a component of the membrane labels in the set L.

Initially, the multiset inside the region consists of n sodium symbols, m symbols of potassium and s symbols of ATP; the multiset from the environment consists of n' sodium symbols and m' symbols of potassium, while the bilayer does not contain any symbols.

Denoting by $R_{Na} = \dfrac{n'}{n}$, $R_K = \dfrac{m'}{m}$ the ratios of occurrences of sodium and potassium ions outside and inside the membrane at any given step, we use this values to describe the starting time for the functioning of the pump. We assume that the activation of the pump is triggered by a change in the values of the ratios evaluated at the current step. Once the following two conditions $R_{Na} > k_1$ and $R_K > k_2$ (for some fixed $k_1, k_2 \in \mathbb{R}$) are satisfied the pump is activated. In [3] a description of the pump using membrane systems is as follows:

Table 5: *The Membrane Systems Model*

	$Env[Bilayer \mid Reg \mid Bilayer]\ Env$
r_1	$: [|_{E_1} Na^3 \overset{(R_{Na}>k_1)\wedge(R_K>k_2)}{\longrightarrow} [Na^3|_{E_1}$
r_2	$: [Na^3|_{E_1} ATP \rightarrow [Na^3|_{E_1^P} ADP$
r_3	$: [Na^3|_{E_1^P} \rightarrow Na^3[|_{E_2^P}$
r_4	$: K^2[|_{E_2^P} \rightarrow [K^2|_{E_2^P}$
r_5	$: [K^2|_{E_2^P} \rightarrow [K^2|_{E_1} P$
r_6	$: [K^2|_{E_1} \rightarrow [|_{E_1} K^2$

3.2 Modelling the Pump with Typed Membrane Systems

The motivation for introducing a type system for membrane systems with symport/antiport rules, namely the class used to model the sodium-potassium pump, comes from the fact that we would like to increase the control in the evolution of the pump. This would mean that if we had a larger set of rules used in the description of the pump, only the ones assuring a correct evolution with respect to the restrictions imposed by the environment would be applied. In this way we increase the control over the evolution of the membrane system.

For the case of the pump we consider the following typing type environment:

$$\Gamma = Na : \mathbf{Na}, K : \mathbf{K}, P : \mathbf{P}, ATP : \mathbf{ATP}, ADP : \mathbf{ADP},$$
$$skin : \{\{\mathbf{Na}, \mathbf{K}\}, \emptyset, \emptyset, \emptyset\}, depth1 : \{\{\mathbf{Na}, \mathbf{K}\}, \{\mathbf{Na}, \mathbf{Na}, \mathbf{Na}\}, \{\mathbf{K}, \mathbf{K}\}, \emptyset\},$$
$$depth2 : \{\{\mathbf{Na}, \mathbf{K}, \mathbf{P}, \mathbf{ATP}, \mathbf{ADP}\}, \{\mathbf{Na}, \mathbf{Na}, \mathbf{Na}\}, \{\mathbf{K}, \mathbf{K}\}, \{E_1, E_2, E_1^P, E_2^P\}\}$$

For the membrane configuration:

$$[K \ldots Na \ldots [[K \ldots Na \ldots ATP]^{E_1}_{depth2}]_{depth1}]_{skin}$$

and the environment Γ defined above, we have that

Lemma 2. $\Gamma \vdash skin : \{S_{skin}, D^{\uparrow}_{skin}, D^{\downarrow}_{skin}, L_{skin}\}$.

Proof.

$$\frac{\dfrac{K : \mathbf{K} \in \Gamma}{\Gamma \vdash K : \mathbf{K}} \quad \dfrac{Na : \mathbf{Na} \in \Gamma}{\Gamma \vdash Na : \mathbf{Na}} \quad depth2 : \{S_{depth2}, D^{\uparrow}_{depth2}, D^{\downarrow}_{depth2}, L_{depth2}\} \in \Gamma \atop \{\mathbf{K}, \mathbf{Na}\} \subseteq S_{depth2}\ E_1 \in L_{depth2} \quad [K \ldots Na \ldots]^{E_1}_{depth2}}{\Gamma \vdash depth2 : \{S_{depth2}, D^{\uparrow}_{depth2}, D^{\downarrow}_{depth2}, L_{depth2}\}}$$

$$\frac{\Gamma \vdash depth2 : \{S_{depth2}, D^{\uparrow}_{depth2}, D^{\downarrow}_{depth2}, L_{depth2}\} \atop depth1 : \{S_{depth1}, D^{\uparrow}_{depth1}, D^{\downarrow}_{depth1}, L_{depth1}\} \in \Gamma \quad [[K \ldots Na \ldots]^{E_1}_{depth2}]_{depth1}}{\Gamma \vdash depth1 : \{S_{depth1}, D^{\uparrow}_{depth1}, D^{\downarrow}_{depth1}, L_{depth1}\}}$$

$$\frac{\dfrac{K : \mathbf{K} \in \Gamma}{\Gamma \vdash K : \mathbf{K}} \quad \dfrac{Na : \mathbf{Na} \in \Gamma}{\Gamma \vdash Na : \mathbf{Na}} \quad skin : \{S_{skin}, D^{\uparrow}_{skin}, D^{\downarrow}_{skin}, L_{skin}\} \in \Gamma \atop {\{\mathbf{K}, \mathbf{Na}\} \subseteq S_{skin} \quad \Gamma \vdash depth1 : \{S_{depth1}, D^{\uparrow}_{depth1}, D^{\downarrow}_{depth1}, L_{depth1}\} \atop [K \ldots Na \ldots [[K \ldots Na \ldots ATP]^{E_1}_{depth2}]_{depth1}]_{skin}}}{\Gamma \vdash skin : \{S_{skin}, D^{\uparrow}_{skin}, D^{\downarrow}_{skin}, L_{skin}\}}$$

The evolution rules from Table 6 state the conditions which must be satisfied for a rule that describes the evolution of the pump to be applied correctly.

Table 6: *Typed Evolution Rules for Pump*

$$\frac{\mathbf{Na} \in S_{depth1} \quad \{\mathbf{Na},\mathbf{Na},\mathbf{Na}\} \in D^{\uparrow}_{depth2}}{[^{E_1}_{depth2}\mathbf{Na}^3 \rightarrow \mathbf{Na}^3[^{E_1}_{depth2}}\,\textbf{(T1)}$$

$$\frac{\mathbf{ADP} \in S_{depth2} \quad E_1^P \in L_{depth2}}{\mathbf{Na}^3[^{E_1}_{depth2}\mathbf{ATP} \rightarrow \mathbf{Na}^3[^{E_1^P}_{depth2}\mathbf{ADP}}\,\textbf{(T2)}$$

$$\frac{\mathbf{Na} \in S_{skin} \quad E_2^P \in L_{depth2} \quad \{\mathbf{Na},\mathbf{Na},\mathbf{Na}\} \in D^{\uparrow}_{depth1}}{[_{depth1}\mathbf{Na}^3[^{E_1^P}_{depth2} \rightarrow \mathbf{Na}^3[_{depth1}[^{E_2^P}_{depth2}}\,\textbf{(T3)}$$

$$\frac{\mathbf{K} \in S_{depth1} \quad \{\mathbf{K},\mathbf{K}\} \in D^{\downarrow}_{depth1}}{\mathbf{K}^2[_{depth1}[^{E_2^P}_{depth2} \rightarrow [_{depth1}\mathbf{K}^2[^{E_2^P}_{depth2}}\,\textbf{(T4)}$$

$$\frac{\mathbf{P} \in S_{depth2} \quad E_1 \in L_{depth2}}{\mathbf{K}^2[^{E_2^P}_{depth2} \rightarrow \mathbf{K}^2[^{E_1}_{depth2}P}\,\textbf{(T5)}$$

$$\frac{\mathbf{K} \in S_{depth2} \quad \{\mathbf{K},\mathbf{K}\} \in D^{\downarrow}_{depth2}}{\mathbf{K}^2[^{E_1}_{depth2} \rightarrow [^{E_1}_{depth2}\mathbf{K}^2}\,\textbf{(T6)}$$

In (**T1**), writing the rule using types, namely $[^{E_1}_{depth2}\mathbf{Na}^3 \rightarrow \mathbf{Na}^3[^{E_1}_{depth2}$ we indicate that any three objects of type **Na** can pass through membrane *depth*2.

Remark 1. Types are used to eliminate (statically) programs in which problems could appear during execution. In the framework of P systems types are used to increase the control, and in this way assuring that no typing problem appears during the evolution of the membrane system. As a consequence, all the ill-typed rules could be eliminated, and the description of the system could be simplified. For example, let us consider the membrane *depth*2 which appears in the typed description of the pump with the type

$$depth2 : \{S_{depth2}, D^{\uparrow}_{depth2}, D^{\downarrow}_{depth2}, L_{depth2}\}$$

where $S_{depth2} = \{\mathbf{Na},\mathbf{K},\mathbf{P},\mathbf{ATP},\mathbf{ADP}\}$, $D^{\uparrow}_{depth2} = \{\mathbf{Na},\mathbf{Na},\mathbf{Na}\}$, $D^{\downarrow}_{depth2} = \{\mathbf{K},\mathbf{K}\}$ and $L_{depth2} = \{E_1, E_2, E_1^P, E_2^P\}$.

Using this typing for *depth*2 membrane, a rule of the form:

$$\mathbf{K}[^{E_1}_{depth2} \rightarrow [^{E_1}_{depth2}\mathbf{K}$$

would be rejected as ill typed since membrane *depth*2 contains in $D^{\downarrow}_{depth2}$ only tuples of two elements of type **K**, so it does not allow single elements of type **K** to be sent inside it. In a similar manner, all the rules which do not satisfy the requirements of the environment are rejected as ill-typed.

4 Conclusion and Future Work

The novelty of this paper is that we introduce types over P systems. In fact we enrich the symport/antiport P systems with typing rules that help to control the evolution of P systems. According to these typing rules, for the typed symport/antiport P systems we prove that if a system is well-typed and an evolution rule is applied, then the obtained system is also well-typed. Another contribution of the paper is the introduction of a type inference algorithm for symport/antiport P systems for which soundness and completeness are proved. We use types in the description of the sodium-potassium pump. This pump was modelled previously using untyped π-calculus [4] and untyped P systems [3].

Our attempt to define a type system for P systems is the first of this kind, and aims to control the evolution of P systems by using types. The type systems can be used in defining generalized rules for P system. For example, by considering a set of typed objects $V = \{X_1 : \mathbf{N_1},\ X_2 : \mathbf{N_1},\ X_3 : \mathbf{N_1},\ A : \mathbf{N_2}\}$ where N_1 and N_2 are some basic types, the evolution rules of the form $X_i \to X_j$, $X_j \to A$, $1 \le i, j \le 3$ can be replaced by rules of a more general form:

1. $\mathbf{N_1} \to \mathbf{N_1}$ (any object of type $\mathbf{N_1}$ can evolve in any object of type $\mathbf{N_1}$);
2. $\mathbf{N_1} \to \mathbf{N_2}$ (any object of type $\mathbf{N_1}$ can evolve in any object of type $\mathbf{N_2}$).

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Molecular Biology of the Cell*, 5th Edition. Garland Science, Taylor & Francis Group, 2008.
2. B. Aman, M. Dezani-Ciancaglini, A. Troina. Type Disciplines for Analysing Biologically Relevant Properties. *Electronic Notes in Theoretical Computer Science*, vol. 227, 97–111, 2009.
3. D. Besozzi, G. Ciobanu. A P System Description of the Sodium-Potassium Pump. *Lecture Notes in Computer Science*, vol. 3365, Springer, 210–223, 2005.
4. G. Ciobanu, V. Ciubotariu, B. Tanasa. A π-Calculus Model of the Na-K Pump. *Genome Informatics*, 469–472, Universal Academy Press, Tokyo, 2002.
5. G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez. *Application of Membrane Computing.* Springer, 2006.
6. F. Fages, S. Soliman. Abstract Interpretation and Types for Systems Biology. *Theoretical Computer Science*, vol. 403, 52–70, 2008.
7. F. Nielson, H. Riis-Nielson, C. Priami, D. Rosa. Control Flow Analysis for BioAmbients. *Electronic Notes in Theoretical Computer Science*, vol. 180, 65–79, 2007.
8. Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.
9. J. Wells. The Essence of Principal Typings. *Lecture Notes in Computer Science*, vol. 2380, Springer, 913–925, 2002.
10. Web page of the P systems: `http://ppage.psystems.eu`.