Life-Death Ratio Approach by a Multiset-Based Type System

Bogdan Aman, Gabriel Ciobanu

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.11, 700506 Iaşi, Romania
baman@iit.tuiasi.ro, gabriel@info.uaic.ro

Summary. We introduce and study a multiset-based type system with ratio thresholds motivated by an important regulatory mechanism inside a cell which try to maintain a "life-death" ratio between some given lower and upper thresholds. We use such a type system to control ratio thresholds in a bio-inspired and multisets-based formalism. For this type system we prove a subject reduction theorem, together with soundness and completeness theorems. A type inference for deducing the type of a system is presented.

1 Introduction

Membrane systems have been introduced as a computational model inspired by cellular biology [9], and have been later applied to the description of biological systems [6]. Possible links between process calculi and membrane systems are presented in [7].

Membrane systems usually consider cells as mechanisms working in a maximal parallel and non-deterministic manner.

However, the living cells do not work in such ways; a chemical reaction takes place only if certain constraints are fulfilled (e.g, certain ratios are between given thresholds in sodium/potassium pump [3] and ratio-dependent predatorprey systems [8]). In order to cope with such constraints, in [2] we enriched the membrane systems with integral proteins by adding a quantitative type discipline. We associated to each system a set of constraints that must be satisfied in order to assure that the application of the rules to a well-formed membrane system leads to a well-formed membrane system as well. We think that this two-stage approach to the description of biological behaviours is of interest, where the first describes reactions in an "untyped" setting, and then rules out certain evolutions by imposing thresholds. This allows one to treat separately different aspects of the modelling: what transitions are possible at all, and under which circumstances they can take place. In this paper we provide a type inference algorithm for deducing the type of a system.

The type system for membrane systems with integral proteins follows the research line started in [1] where a type system for membrane system with symport/antiport rules is presented. The work is also related to [4] where a type systems for the calculus of looping sequences is defined based on the number of elements and not on the ratios between elements. The presentation of the typed sodium/potassium pump in [2] is used as a motivation and a running example for typing the membrane systems. This paper is related to our previous article [2].

2 Membrane Systems with Integral Proteins

Membrane systems are parallel and nondeterministic computing models inspired by the compartments of cells and 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 region called the *skin* membrane. Multisets of objects are distributed inside these regions, and they can be modified or moved between adjacent compartments. Objects represent the formal counterpart of the molecular species (ions, proteins, etc.) floating inside cellular compartments, and they are described by means of strings over a given alphabet. Evolution rules represent the formal counterpart of chemical reactions, and are given in the form of rewriting rules that operate on the objects, as well as on the compartmentalised structure (e.g., by dissolving, dividing, creating, or moving membranes).

Starting from an initial configuration, the multisets of objects initially placed inside the compartmentalised structure, the system evolves by applying the evolution rules in a nondeterministic and maximally parallel manner. A rule is applicable when all the objects appearing on its left hand side are available in the region where the rule is placed. The maximal parallelism of rule application means that each applicable rule that is inside a region *has to* be applied in that region. Since there is a competition for the available objects, only certain (nondeterministically selected) rules are applied. A halting configuration is reached when no rule is applicable, and the result is given by the number of objects (in a specified region). More details can be found in [9, 10].

In what follows we present some technical notions used in this paper. Given a finite set O of symbols, the set of all strings over O is denoted by O^*, and the set of all non-empty strings over O, is denoted by $O^+ = O^* \backslash \lambda$, where λ is the empty string. A multiset over O is a map $u : O \to \mathbb{N}$, where $u(a)$ denotes the multiplicity of the symbol $a \in O$ in the multiset u and $|u| = \sum_{a \in O} u(a)$ denotes the number of objects appearing in the multiset u. We say that a multiset u is included into a multiset v (denoted by $u \subseteq v$) if $u(a) \leq v(a)$ for all $a \in O$. The empty multiset is denoted by ϵ, and satisfies $\epsilon(a) = 0$, for all $a \in O$. For two multisets u and v we define the sum $u + v$ by $(u + v)(a) = u(a) + v(a)$ for all $a \in O$, and the difference $u - v$ by $(u - v)(a) = max\{0, u(a) - v(a)\}$ for all $a \in O$. More details can be found in [11].

Inspired by [5], we present a membrane system that is able to model biological pumps, in which exist attachment/de-attachment of objects to/from the integral proteins of the membranes, and transformation of objects inside a region if certain integral proteins are present in the surrounding membranes. To each membrane is associated a label $i \in Lab$, and two multisets s_i and v_i over O^*, and the membrane is denoted by $[\ s_i\ ]^i_{v_i}$. If s_i and/or v_i are the empty multisets, they are omitted.

Definition 1. *A* membrane system with integral proteins of degree n *is:*

$$\prod = (O, Lab, \mu, v_1/s_1, \ldots, v_n/s_n, R),\ \textit{where:}$$

1. O *is an alphabet (finite, non-empty) of objects;*
2. L *is a finite set of labels;*
3. μ *is a membrane hierarchical structure with* $n \geq 2$ *membranes;*
4. v_i, $1 \leq i \leq n$ *is a multiset of integral proteins present on the membrane* i*;*
5. s_i, $1 \leq i \leq n$ *is a multiset of objects placed inside the membrane* i*;*
6. R *is a finite set of rules of the following forms:*

a) $[\ \alpha\ ]^i_v \rightarrow [\ ]^i_{v'}$, $\alpha \in O^+$, $v, v' \in O^*$, $i \in Lab$; $(attach_{in})$

This rule is applicable if the following conditions are fulfilled:

- *the multiset* α *exists inside region* i $(\alpha \subseteq s_i)$*;*
- *the multiset* v *of integral proteins exists on membrane* i $(v \subseteq v_i)$*.*

When this rule is applied to membrane i*:*

- α *is removed from* s_i*;*
- v *is modified to* v'*;*
- *the objects not involved in this rule are left unchanged.*

b) $[\ [\ ]^i_v\ \alpha]^j \rightarrow [\ [\ ]^i_{v'}]^j$, $\alpha \in O^+$, $v, v' \in O^*$, $i, j \in Lab$; $(attach_{out})$

This rule is applicable if the following conditions are fulfilled:

- *the multiset* α *exists inside region* j $(\alpha \subseteq s_j)$*;*
- *the multiset* v *of integral proteins exists on membrane* i $(v \subseteq v_i)$*.*

When this rule is applied to membrane i *we have:*

- α *is removed from* s_j*;*
- v *is modified to* v'*;*
- *the objects not involved in this rule are left unchanged.*

c) $[\ ]^i_v \rightarrow [\ \alpha\ ]^i_{v'}$, $\alpha \in O^+$, $v \in O^+$, $v' \in O^*$, $i \in Lab$; $(de-attach_{in})$

This rule is applicable if the following conditions are fulfilled:

- *the multiset* v *of integral proteins exists on membrane* i $(v \subseteq v_i)$*.*

When this rule is applied to membrane i*:*

- v *is modified to* v'*;*
- α *is added to* s_i*;*
- *the objects not involved in this rule are left unchanged.*

d) $[\ [\ ]^i_v]^j \rightarrow [\ [\ ]^i_{v'}\ \alpha]^j$, $\alpha \in O^+$, $v \in O^+$, $v' \in O^*$, $i, j \in Lab$; $(de-attach_{out})$

This rule is applicable if the following conditions are fulfilled:

- *the multiset* v *of integral proteins exists on membrane* i $(v \subseteq v_i)$*.*

When this rule is applied to membrane i *we have:*

- v *is modified to* v'*;*

- *α is added to s_j;*
- *the objects not involved in this rule are left unchanged.*

e) $[\ \alpha\ ]^i_v \to [\beta\ ]^i_v$, $v, \beta \in O^*$, $\alpha \in O^+$, *and* $i \in Lab$. (*local evol*)

This rule is applicable if the following conditions are fulfilled:

- *the multiset α exists inside region i ($\alpha \subseteq s_i$);*
- *the multiset v of integral proteins exists on membrane i ($v \subseteq v_i$).*

When this rule is applied to membrane i:

- *α is removed from s_i;*
- *β is added to s_i;*
- *the objects not involved in this rule are left unchanged.*

The way in which the evolution rules are applied is detailed in what follows. In order to formally represent the configurations of membrane systems with integral proteins, we define terms ranged over by st, st_1, ..., that are built by means of a membrane constructor $[-]^-_-$, using a set O of objects and a set Lab of labels. The syntax of the terms $st \in ST$ is given by

$$st ::= u \mid [st]^i_v \mid st\ st\ ,$$

where u denotes a (possibly empty) multiset of objects placed inside a membrane, v a multiset of objects within or on the surface of a membrane, i a membrane label, and $st\ st$ is the parallel composition of two terms. Since we work with multisets of terms, we introduce a structural congruence relation following a standard approach from process algebra. The defined structural congruence is the least congruence relation on terms satisfying also the rule:

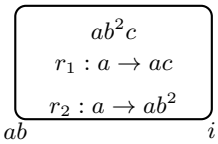
$$\text{if } v_1 \equiv v_2 \text{ and } st_1 \equiv st_2 \text{ then } [st_1]^i_{v_1} \equiv [st_2]^i_{v_2}.$$

A pattern is a term that may include variables. We denote by $\mathcal{P}$ the infinite set of patterns P of the form: $P ::= st \mid [\ P\ X\ ]^i_{v\ y} \mid P\ P$. We distinguish between "simple variables" (ranged over by x, y, z) that may occur only on the surface of membranes (i.e., they can be replaced only by multisets of objects) and "term variables" (ranged over by X, Y, Z) that may only occur inside regions (they can be replaced by arbitrary terms). Therefore, we assume two disjoint sets: V_{O^*} (set of simple variables) and V_{ST^*} (set of term variables). We denote by $V = V_{O^*} \cup V_{ST^*}$ the set of all variables, and with ρ any variable in V.

An instantiation is a partial function $\sigma : V \to ST^*$ that preserves the type of all variables: simple variables ($x \in V_{O^*}$) and term variables ($X \in V_{ST^*}$) are mapped into objects ($\sigma(x) \in O^*$) and terms ($\sigma(X) \in ST^*$), respectively. Given a pattern P, the term obtained by replacing all occurrences of each variable $\rho \in V$ with the term $\sigma(\rho)$ is denoted by $P\sigma$. The set of all possible instantiations is denoted by Σ, and the set of all variables appearing in P is denoted by $Var(P)$.

Formally, a rewriting rule r is a pair of patterns (P_1, P_2), denoted by $P_1 \to P_2$, where $P_1 \neq \epsilon$ (i.e., P_1 is a non-empty pattern) and $Var(P_2) \subseteq Var(P_1)$. A rewriting rule $P_1 \to P_2$ states that a term $P_1\sigma$ can be transformed into the term $P_2\sigma$, for some instantiation function σ.

Example 1. Consider the membrane system depicted in what follows. In the right part of the picture we give some examples of the notions defined above.



Terms: $[ab^2c]^i_{ab}$ or ab^2c
Patterns: $[aX]^i_y$ or aX
Instantiation: $\sigma(X) = b^2c$, $\sigma(y) = ab$
Rewriting rule r_1: $aX \to acX$
Rewriting rule r_2: $aX \to ab^2X$

It can be noticed that using the given instantiations the pattern $[aX]^i_y$ becomes the term $[ab^2c]^i_{ab}$, while the pattern aX becomes the the term ab^2c. The rewriting rules are a generalization of the rules used in P systems, by adding variables that stand for the objects not involved in the evolution of a system.

The notion of context is used to complete the definition of a rewriting semantics for our systems. This is done by enriching the syntax with a new object $\Box$ representing a hole. By definition, a context is represented as a single hole $\Box$. The infinite set $\mathcal{C}$ of contexts (ranged over by C) is given by:

$$C ::= \Box \mid C\ st \mid [\ C\ ]^i_v.$$

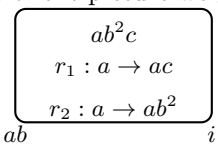
Given $C_1, C_2 \in \mathcal{C}$, $C_1[\ st\ ]$ denotes the term obtained by replacing $\Box$ with st in C_1, while $C_1[\ C_2\ ]$ denotes the context obtained by replacing $\Box$ with C_2 in C_1.

Given a set R of rewriting rules, a reduction semantics of the system is given by the least transition relation $\to$ closed with respect to $\equiv$ satisfying also the rule:

$$\frac{P_1 \to P_2 \in R \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma \quad C \in \mathcal{C}}{C[P_1\sigma] \to C[P_2\sigma]}\ .$$

$\to^*$ denotes the reflexive and transitive closure of $\to$.

Example 2. Consider the membrane system depicted in what follows. In the right part of the picture we give some examples of the notions defined previously.



Rewriting rule r_1: $aX \to acX$
Instantiation: $\sigma(X) = b^2c$
Context: $[\Box]^i_{ab}$
Transition: $[ab^2c]_{ab} \to [ab^2c^2]_{ab}$

It can be noticed that using the given instantiation and context the rewriting rule r_1: $aX \to acX$ can be used to perform the transition $[ab^2c]_{ab} \to [ab^2c^2]_{ab}$, thus modelling the application of rule r_1 is the above membrane system.

3 Threshold-Based Type System Over Multisets

We use the type system defined in [2]. Let T be a finite set of basic types ranged over by t. We classify each object in O with a unique element of T; we use Γ to denote this classification. In general, different objects a and b can have the same basic type t. When there is no ambiguity, we denote the type associated with an object a by t_a. For each ordered pair of basic types (t_1, t_2), we assume the existence of two functions, $min : T \times T \to (0, \infty) \cup \{\diamond\}$ and $max : T \times T \to (0, \infty) \cup \{\diamond\}$. These functions indicate the minimum and maximum ratio between the number of objects of basic types t_1 and t_2 that can be present inside a membrane.

Definition 2 (Consistent Basic Types). *A system using a set of basic types T and the functions min and max is consistent if:*

1. $\forall t_1, t_2 \in T$, $min(t_1, t_2) \neq \diamond$ *iff* $max(t_1, t_2) \neq \diamond$;
2. $\forall t_1, t_2 \in T$, $min(t_1, t_2) \neq \diamond$ *iff* $min(t_2, t_1) \neq \diamond$;
3. $\forall t_1, t_2 \in T$ *if* $min(t_1, t_2) \neq \diamond$, *then* $min(t_1, t_2) \leq max(t_1, t_2)$;
4. $\forall t_1, t_2 \in T$ *if* $min(t_1, t_2) \neq \diamond$ *and* $max(t_2, t_1) \neq \diamond$, *then* $min(t_1, t_2) \cdot max(t_2, t_1) = 1$.

Example 3. Consider $T = \{t_a, t_b, t_c\}$ and min, max defined as:

$min(t_1, t_2)$

$t_1 \backslash t_2$	t_a	t_b	t_c
t_a	$\diamond$	0.4	$\diamond$
t_b	0.2	$\diamond$	1/6
t_c	$\diamond$	3	$\diamond$

$max(t_1, t_2)$

$t_1 \backslash t_2$	t_a	t_b	t_c
t_a	$\diamond$	5	$\diamond$
t_b	2.5	$\diamond$	1/3
t_c	$\diamond$	6	$\diamond$

The system is consistent since each pair of types respects Definition 2.

Definition 3. Quantitative types *are triples (L, Pr, U) over the set T of basic types, where L (lower) is the set of minimum ratios between basic types, Pr (present) is the multiset of basic types of present objects (the objects present at the top level of a pattern, i.e. in the outermost membrane), and U (upper) is the set of maximum ratios between basic types.*

The number of objects of type t appearing in a multiset Pr is denoted by $Pr(t)$. In order to define well-formed types, given a multiset M of types, the sets RP_M (ratios of present types in M), L_M (lower bounds of present types in M) and U_M (upper bounds of present types in M) are required:

- $RP_M = \begin{cases} \emptyset & \text{if } |M| \leq 1 \\ \bigcup_{t,t' \in M} \left\{ t/t' : \dfrac{Pr(t)}{Pr(t')} \mid t \neq t', Pr(t') \neq 0 \right\} & \text{otherwise} \end{cases}$
- $L_M = \begin{cases} \emptyset & \text{if } |M| \leq 1 \\ \bigcup_{t,t' \in M} \{ t/t' : min(t, t') \mid t \neq t', min(t, t') \neq \diamond \} & \text{otherwise} \end{cases}$
- $U_M = \begin{cases} \emptyset & \text{if } |M| \leq 1 \\ \bigcup_{t,t' \in M} \{ t/t' : max(t, t') \mid t \neq t', max(t, t') \neq \diamond \} & \text{otherwise} \end{cases}$

These sets contain labelled values in order to be able to refer to them when needed: e.g., $t/t' : \dfrac{Pr(t)}{Pr(t')}$ denotes the fact that the ratio between the objects of types t and t' that are present in Pr has the label t/t', and the value is $\dfrac{Pr(t)}{Pr(t')}$.

Definition 4 (Well-Formed Types). *A type (L, Pr, U) is well-formed if*
$$L = L_{Pr},\ U = U_{Pr} \text{ and } L \leq RP_{Pr} \leq U.$$

Remark 1. If the set T contains a large number of basic types, defining a type to be well-formed only if $L = L_{Pr}$ and $U = U_{Pr}$ reduces the amount of information encapsulated by a type. E.g., for $|T| = 100$, the number of entries in the *min* table is equal to 10000.

Example 4. Let us assume a set of basic types $T = \{t_a, t_b\}$, a classification $\Gamma = \{a : t_a, b : t_b\}$ and the functions *min*, *max* defined as:

$min(t_1, t_2)$

$t_1 \backslash t_2$	t_1	t_2
t_1	$\diamond$	0.5
t_2	0.3	$\diamond$

$max(t_1, t_2)$

$t_1 \backslash t_2$	t_1	t_2
t_1	$\diamond$	10/3
t_2	2	$\diamond$

The term a^5b^2 is well-formed, while the term a^9b is not, because the ratio between t_a and t_b equals 9, and so it exceeds the maximum 10/3 indicated in *max* table.

From now on we work only with well-formed types. For instance, the two well-formed types (L, Pr, U) and (L', Pr', U') of the following two definitions are constructed by using specific ratio tables for *min* and *max*.

Definition 5 (Type Compatibility). *Two well-formed types* (L_{Pr}, Pr, U_{Pr}) *and* $(L_{Pr'}, Pr', U_{Pr'})$ *are compatible, written* $(L, Pr, U) \bowtie (L_{Pr'}, Pr', U_{Pr'})$, *if* $L_{Pr+Pr'} \leq RP_{Pr+Pr'} \leq U_{Pr+Pr'}$.

Example 5 (cont.). Consider the assumptions from Example 4. The types $(L_{t_a^5\ t_b^2}, t_a^5 t_b^2, U_{t_a^5\ t_b^2})$, $(\emptyset, t_a, \emptyset)$ and $(\emptyset, t_a^4, \emptyset)$ are well-formed. It holds that

$$(L_{t_a^5 t_b^2}, t_a^5 t_b^2, U_{t_a^5\ t_b^2}) \bowtie (\emptyset, t_a, \emptyset) \text{ and } (L_{t_a^5 t_b^2}, t_a^5\ t_b^2, U_{t_a^5\ t_b^2}) \not\bowtie (\emptyset, t_a^4, \emptyset),$$

since the ratio in the second case between t_a and t_b equals 4.5, and so it exceeds the maximum 10/3 indicated in *max* table.

A **basis** Δ assigning types to simple and term variables is defined by

$$\Delta ::= \emptyset \mid \Delta, x : (L_t, t, U_t) \mid \Delta, X : (L, Pr, U).$$

A basis is well-formed if all types in the basis are well-formed.

A **classification** Γ maps each object in O to a unique element of the set T of basic types. The judgements are of the form $\Delta \vdash P : (L, Pr, U)$ indicating that a pattern P is well-typed having the type (L, Pr, U) relative to a typing environment Δ.

Types are assigned to patterns and terms according to the typing rules of Table 1. It is not difficult to verify that a derivation starting from well-formed bases produces only well-formed bases and well-formed types.

We define a typed semantics, since we are interested in applying reduction rules only to correct terms having well-formed types, and whose requirements are satisfied. More formally, a term *st* is correct if $\emptyset \vdash st : (L, Pr, U)$ for some well-formed type (L, Pr, U). An instantiation σ agrees with a basis Δ (denoted by $\sigma \in \Sigma_\Delta$) if $\rho : (L, Pr, U) \in \Delta$ implies $\emptyset \vdash \sigma(\rho) : (L, Pr, U)$.

In order to apply the rules in a safe way, we introduce a restriction on rules based on the context of application rather than on the type of patterns involved in

Table 1. Typing Rules

$$\Delta \vdash \epsilon : (\emptyset, \emptyset, \emptyset) \quad (TEps) \qquad \frac{a : t \in \Gamma}{\Delta \vdash a : (L_t, t, U_t)} \quad (TObj)$$

$$\Delta, \rho : (L, Pr, U) \vdash \rho : (L, Pr, U) \quad (TVar)$$

$$\frac{\Delta \vdash v : (L, Pr, U) \quad \Delta \vdash P' : (L', Pr', U') \quad (L, Pr, U) \bowtie (L', Pr', U') \quad i \in Lab}{\Delta \vdash [P']_v^i : (L, Pr, U)} \quad (TMem)$$

$$\frac{\Delta \vdash P : (L_{Pr}, Pr, U_{Pr}) \quad \Delta \vdash P' : (L_{Pr'}, Pr', U_{Pr'}) \quad (L_{Pr}, Pr, U_{Pr}) \bowtie (L_{Pr'}, Pr', U_{Pr'})}{\Delta \vdash P\ P' : (L_{Pr+Pr'}, Pr + Pr', U_{Pr+Pr'})} \quad (TPar)$$

the rule. In this direction, we characterise contexts by the types of terms that can fill their hole, and the rules by the types of terms produced by their application.

Definition 6 (Typed Holes). *Given a context C and a type (L, Pr, U) that is well-formed, the type (L, Pr, U)* **fits the context** *C if for some well-formed type (L', Pr', U') it can be shown that $X : (L, Pr, U) \vdash C[X] : (L', Pr', U')$.*

Example 6 (cont.). Consider the notions from Example 4, and also

- a term $[a^5\ b^4]a^{12}\ b^3$,
- a rule $r_1 : [\ a^3\ X\ ]_x \to [\ X\ ]_{a^3\ x}$ (having the form $P_1 \to P_2$),
- a context $C = \Box\ a^{12}\ b^3$,
- instantiations $\sigma(X) = a^2\ b^4$, $\sigma(x) = \epsilon$.

By applying rule r_1 the term $[a^2\ b^4]_{a^3}a^{12}\ b^3$ is obtained. This is not well-typed since the type $(L_{t_a^3}, t_a^3, U_{t_a^3})$ of P_2 does not fit the context C. It does not fit since the term $C[P_2]$ has the type $(L_{t_a^{15}t_b^3}, t_a^{15}t_b^3, U_{t_a^{15}t_b^3})$ that is not well-formed due to the fact that the ratio between t_a and t_b at the top level is $\frac{12+3}{3} = 5$ that is greater than $10/3$.

The above notion guarantees that we obtain a correct term filling a context with a term whose type fits the context: note that there may be more than one type (L, Pr, U) such that (L, Pr, U) fits the context C.

We can classify reduction rules according to the types that can be derived for the right hand sides of the rules, since they influence the type of the obtained term.

Definition 7 (Δ-(L, Pr, U) safe rules). *A rewriting rule $P_1 \to P_2$ is Δ safe if for some well-formed type (L, Pr, U) it can be shown that $\Delta \vdash P_2 : (L, Pr, U)$.*

To ensure correctness, each application of a rewriting rule must verify that the type of the right hand side of the rule fits the context. Using Definitions 6 and 7, if it is applied a rule whose right hand side has type (L, Pr, U) and this type fits the context, then a correct term is obtained.

Typed Semantics.

Given a finite set R of rewriting rules (e.g., the one presented in Table 2), the typed semantics of a system is given by the least relation $\Rightarrow$ closed with respect to $\equiv$ and satisfying the following rule:

$$\frac{\begin{array}{c} P_1 \to P_2 \in R \text{ is a } \Delta\text{-}(L, Pr, U) \text{ safe rule, } P_1\sigma \not\equiv \epsilon \\ \sigma \in \Sigma_\Delta \qquad C \in \mathcal{C} \qquad \text{and} \qquad (L, Pr, U) \text{ fits } C \end{array}}{C[P_1\sigma] \Rightarrow C[P_2\sigma]} \qquad (TSem)$$

Using weakening and substitution properties proved in [2], we can provide the result that well-formed bases guarantees type preservation.

Proposition 1. *For all $\sigma \in \Sigma_\Delta$, $\emptyset \vdash P\sigma : (L, Pr, U)$ iff $\Delta \vdash P : (L, Pr, U)$.*

Starting from a correct term, all the terms obtained via Δ-(L, Pr, U) safe rules are correct, and thus avoiding conditions over P_1 as they do not influence the type of the obtained term. As desired, typed reduction preserves correctness.

Theorem 1 (Subject Reduction). *If $\emptyset \vdash st : (L, Pr, U)$ and $st \Rightarrow st'$, then $\emptyset \vdash st' : (L', Pr', U')$ for a well-formed type (L', Pr', U').*

4 Type Inference

To formalize type reconstruction, we will need a concise way of talking about the possible ways that type variables can be substituted by types, in a term and its associated context, to obtain a valid typing statement.

In this section we define inference rules in order to derive which rules are Δ-(L, Pr, U) safe, where the choices of Δ, L, Pr, U are guided by the initial pattern. Formally, given a typable pattern P, there exists a typing $\Delta \vdash P : (L, Pr, U)$. The typed semantics of rule $(Tsem)$ does not show how to choose Δ and Pr (L and U depend on Pr).

We assume that for each object variable x there is an o-type variable ρ_x ranging over basic types, and for each term variable X there is an m-type variable λ_X ranging over multisets of basic types. Moreover, we assume that Λ ranges over unions of multisets of basic types, o-type and m-type variables.

A basic scheme Θ is a mapping from atomic variables to their o-type variables, and from term variables to triples of their l-type variables, pr-type variables and u-type variables:

$$\Theta ::= \emptyset \mid \Theta, x : \rho_x \mid \Theta, X : \lambda_X.$$

The rules for inferring the typings use judgements of the form:

$$\vdash P : \Theta; (L_\Lambda, \Lambda, U_\Lambda); \Xi$$

where Θ is the basis scheme in which P is well-formed, $(L_\Lambda, \Lambda, U_\Lambda)$ is the type of P, and Ξ is the set of constraints that should be satisfied. Table 2 presents the inference rules, derived from the typing rules of Table 1.

Example 7. Let us assume a set of basic types $T = \{t_a, t_b\}$, a classification $\Gamma = \{a : t_a, b : t_b\}$ and the functions min, max defined as:

$min(t_1, t_2)$

$t_1 \backslash t_2$	t_1	t_2
t_1	$\diamond$	0.6
t_2	0.25	$\diamond$

$max(t_1, t_2)$

$t_1 \backslash t_2$	t_1	t_2
t_1	$\diamond$	4
t_2	5/3	$\diamond$

Consider the well-formed term $a^5\ b^2$. In order to infer its type there are several ways of considering the patterns P_1 and P_2 in order to apply the last $(IPar)$ rule. Consider the following two cases:

- $P_1 = a^3b^2$ (well-formed) and $P_2 = a^2$ (well-formed); in this case the rule $(IPar)$ leads to a well-formed type with all constraints satisfied;
- $P_1 = ab^2$ (not well-formed) and $P_2 = a^4$ (well-formed); in this case the type-compatibility condition is not fulfilled even if the obtained term is well-formed.

In order to obtain the type of the term $a^5\ b^2$, there are considered different decompositions until there is obtained an well-formed type and all conditions hold, or all possible decompositions do not provide a well-formed type with the conditions fulfilled.

Table 2. Inference Rules

$$\vdash \epsilon : \emptyset; (\emptyset, \emptyset, \emptyset); \emptyset \quad (IEps) \qquad \frac{a : t \in \Gamma}{\vdash a : \emptyset; (L_t, t, U_t); \emptyset} \quad (IObj)$$

$$\vdash x : \{x : \rho_x\}; (L_{\rho_x}, \rho_x, U_{\rho_x}); \emptyset \qquad (IVar1)$$

$$\vdash X : \{X : \lambda_X\}; (L_{\lambda_X}, \lambda_X, U_{\lambda_X}); \emptyset \qquad (IVar2)$$

$$\frac{\vdash v : \Theta; (L_\Lambda, \Lambda, U_\Lambda); \Xi \quad \vdash P' : \Theta'; (L_{\Lambda'}, \Lambda', U_{\Lambda'}); \Xi' \quad i \in Lab}{\vdash [P']^i_v : \Theta \cup \Theta'; (L_\Lambda, \Lambda, U_\Lambda); \Xi \cup \Xi' \cup \{(L_\Lambda, \Lambda, U_\Lambda) \bowtie (L_{\Lambda'}, \Lambda', U_{\Lambda'})\}} \quad (IMem)$$

$$\frac{\vdash P : \Theta; (L_\Lambda, \Lambda, U_\Lambda); \Xi \quad \vdash P' : \Theta'; (L_{\Lambda'}, \Lambda', U_{\Lambda'}); \Xi'}{\begin{array}{c}\vdash PP' : \Theta \cup \Theta'; (L_{\Lambda+\Lambda'}, \Lambda + \Lambda', U_{\Lambda+\Lambda'}); \\ \Xi \cup \Xi' \cup \{(L_\Lambda, \Lambda, U_\Lambda) \bowtie (L_{\Lambda'}, \Lambda', U_{\Lambda'})\}\end{array}} \quad (IPar)$$

The rules of Table 2 are easily derived from the rules of Table 1. The basis is the union of the basis of the composing patterns, without renaming, because each variable x or X is associated with an unique o-type variable, or to an unique m-type variable, respectively. The key difference between inference rules of Table 2 and typing rules of Table 1 is that the conditions of type compatibility and type satisfaction are not premises, but conclusions. In this way, at the end of inference, all these conditions create a set of constraints that must be checked to decide the applicability of the rules.

Soundness and completeness of our inference rules can be stated as usual. A type mapping associates o-type variables to basic types, m-type variables to multisets of basic types. A type mapping $\mathbf{m}$ satisfies a set of constraints Ξ if all the constraints in $\mathbf{m}(\Xi)$ hold.

Theorem 2 (Soundness of Type Inference).
If $\vdash P : \Theta; (L_\Lambda, \Lambda, U_\Lambda); \Xi$ *and* $\mathbf{m}$ *is a type mapping satisfying* Ξ*, then*

$$\mathbf{m}(\Theta) \vdash P : (\mathbf{m}(L_\Lambda), \mathbf{m}(\Lambda), \mathbf{m}(U_\Lambda)).$$

Theorem 3 (Completeness of Type Inference). *If* $\Delta \vdash P : (L_{Pr}, Pr, U_{Pr})$*, then* $\vdash P : \Theta; (L_\Lambda, \Lambda, U_\Lambda); \Xi$ *for some* Θ*,* Λ*,* Ξ *such that there is a type mapping* $\mathbf{m}$ *that satisfies* Ξ*,* $\Delta \supseteq \mathbf{m}(\Theta)$*, and* $Pr = \mathbf{m}(\Lambda)$*.*

We use inference rules to decide applicability of typed reduction rules for Δ-(L_{Pr}, Pr, U_{Pr}) safe rules. The first step is to see when a type mapping ensures that a rule is a Δ-(L_{Pr}, Pr, U_{Pr}) safe rule, i.e. when it satisfies the constraints of Definition 7. The concept of Δ-(L_{Pr}, Pr, U_{Pr}) safety is used to classify rules according to the types we can derive for the right hand side patterns of them.

Lemma 1 (Characterisation of Δ-(L, Pr, U) safe rules).
A rule $P_1 \to P_2$ *is* Δ-(L_{Pr}, Pr, U_{Pr}) *safe if and only if the type mapping* $\mathbf{m}$ *defined by the basis* Δ*, i.e. such that*

- $\mathbf{m}(\rho_x) = t$ *if* $\Delta(x) = t$*, and*
- $\mathbf{m}(\lambda_X) = Pr$ *if* $\Delta(X) = (L_{Pr}, Pr, U_{Pr})$*.*

satisfies the set of constraints $\Xi_2 \cup \{\Lambda_2 = Pr\}$*, whenever it holds that* $\vdash P_2 : \Theta_2; (L_{\Lambda_2}, \Lambda_2, \Omega_2); U_{\Lambda_2}$*.*

Since Δ-(L_{Pr}, Pr, U_{Pr}) safe rules can be applied only in contexts in which type (L_{Pr}, Pr, U_{Pr}) fits, we must characterise also the fitting relation.

Lemma 2 (Characterisation of Fitting Relation).
Let the context C *be such that* $\vdash C[T] : (L_{Pr_C}, Pr_C, U_{Pr_C})$ *for some* T *and well-formed type* $(L_{Pr_C}, Pr_C, U_{Pr_C})$*. A well-formed type* (L_{Pr}, Pr, U_{Pr}) *fits* C *if and only if the type mapping* $\mathbf{m}$ *defined by*

$$\mathbf{m}(\lambda_X) = Pr$$

satisfies the set of constraints

$$\Xi_C \cup \{(L_{\Lambda_C}, \Lambda_C, U_{\Lambda_C}) \text{ is well-formed}\},$$

where $\vdash C[X] : \{X : \lambda_X\}; (L_{\Lambda_C}, \Lambda_C, U_{\Lambda_C}); \Xi_C$*.*

Using the characterisation of Δ-(L, Pr, U) safe rules and the fitting relation, we can infer the applicability of a rewriting rule by checking if the type mapping respects the required constraints.

Theorem 4 (Applicability of Rewriting Rules).
If the following conditions are fulfilled
$\vdash P_1 : \Theta_1; (L_{\Lambda_1}, \Lambda_1, U_{\Lambda_1}); \Xi_1$*,*
$\vdash P_2 : \Theta_2; (L_{\Lambda_2}, \Lambda_2, U_{\Lambda_2}); \Xi_2$*,*
$\vdash C[X] : \{X : \lambda_X\}; (L_{\Lambda_C}, \Lambda_C, U_{\Lambda_C}); \Xi_C$
and $P_1\sigma \neq \epsilon$*, then the rule* $P_1 \to P_2$ *can be applied to the term* $C[P_1\sigma]$ *such that* $\vdash C[P_1\sigma] : (L_{Pr_C}, Pr_C, U_{Pr_C})$ *for a well-formed type* $(L_{Pr_C}, Pr_C, U_{Pr_C})$ *if and only if the type mapping* $\mathbf{m}$ *defined by*

- $\mathbf{m}(\rho_x) = t$, *if* $\sigma(x) : t \in \Gamma$,
- $\mathbf{m}(\lambda_X) = Pr$, *if* $\vdash \sigma(X) : (L_{Pr}, Pr, U_{Pr})$

satisfies the set of constraints

$$\Xi_2 \cup \{(\lambda_X = \Lambda_2)\} \cup \Xi_C \cup \{(L_{\Lambda_C}, \Lambda_C, U_{\Lambda_C}) \text{ is well-formed}\}.$$

5 Conclusion

This paper is related to a previous approach presented in [2], where a quantitative types based on ratio thresholds is introduced. The inspiration was the first discovered ion transporter (awarded with a Nobel Prize in 1997), namely the sodium-potassium pump, which extrudes sodium ions in exchange for potassium ions. These exchanges take place only if the ratios of these elements are between certain lower and upper thresholds. To cope properly with such constraints, we introduced in a multiset-based type system with ratio thresholds, where the sodium/potassium pump is used as a running example. We associated to each system a set of constraints, and relate them to the ratios between elements. If the constraints are satisfied, we prove that if a system is well-typed and an evolution rule is applied, then the obtained system is also well-typed.

In this paper we defined a type inference algorithm for membrane systems with integral proteins for which soundness and completeness are proved. The work intends to allow automatic analyse of inference according to the defined type system, and to provide support for the development of software tools.

The proposed typed semantics completely excludes the fact that sometimes biological constraints can be broken leading to a disease or even to the death of the biological system. However, the typed semantics can be modified in order to allow transitions that lead to terms that are not typable. In this case the type system should signal that some undesired event has been reached. In this way, it can be checked if a term breaks some biological property, or if the system has some unwanted behaviour.

Acknowledgements. The work was supported by a grant of the Romanian National Authority for Scientific Research, project number PN-II-ID-PCE-2011-3-0919.

References

1. Aman, B. and Ciobanu, G. Typed Membrane Systems. *Lecture Notes in Computer Science* **5957**, 169–181 (2010).
2. Aman, B. and Ciobanu, G. Behavioural Types Inspired by Cellular Thresholds. *Lecture Notes in Computer Science* **8368**, 1–15 (2014).
3. Besozzi, D. and Ciobanu, G. A P System Description of the Sodium-Potassium Pump. *Lecture Notes in Computer Science* **3365**, 210–223 (2005).
4. Bioglio, L. Enumerated Type Semantics for the Calculus of Looping Sequences. *RAIRO - Theoretical Informatics and Applications* **45** (1), 35–58 (2011).

5. Cavaliere, M. and Sedwards, S. Modelling Cellular Processes Using Membrane Systems With Peripheral and Integral Proteins. *Lecture Notes in Bio-Informatics* **4210**, 108–126 (2006).
6. Ciobanu, G., Păun, Gh. and Pérez-Jiménez, M.J.(Eds.) *Applications of Membrane Computing*. Springer (2006).
7. Ciobanu, G. *Membrane Computing and Biologically Inspired Process Calculi*. "A.I.Cuza" University Press, Iaşi (2010).
8. Hsu, S.-B., Hwang, T.-W. and Kuang, Y. A Ratio-Dependent Food Chain Model and Its Applications to Biological Control. *Mathematical Biosciences* **181**, 55–83 (2003).
9. Păun, Gh. *Membrane Computing. An Introduction*. Springer (2002).
10. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) *Handbook of Membrane Computing*. Oxford University Press (2010).
11. Salomaa, A. *Formal Languages*. Academic Press (1973).
12. Wells, J. The Essence of Principal Typings. *Lecture Notes in Computer Science* **2380**, 913–925 (2002).