On Two Families of Multiset Tree Automata[⋆]

José M. Sempere, Damián López

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
{jsempere,dlopez}@dsic.upv.es

Summary. The relation between the membrane structures of P systems and an extension of tree automata which introduces multisets in the transition function has been proposed in previous works. Here we propose two features of tree automata which have been previously studied (namely, reversibility and local testability) in order to extend them to multiset tree automata. The characterization of these families will introduce a new characterization of membrane structures defined by the set of rules used for membrane creation and deletion.

1 Introduction

The relation between membrane structures and tree languages has been explored in previous works. So, Freund et al. [4] proved that P systems are able to generate recursively enumerable sets of trees through their membrane structures. Other works have focused on extending the definition of finite tree automata in order to take into account the membrane structures generated by P systems. So, in [13], the authors propose an extension of tree automata, namely multiset tree automata, in order to recognize membrane structures. In [7], the authors propose the use of this model to calculate editing distances between membrane structures. Later, the authors proposed a method to infer multiset tree automata from membrane observations [14].

In this work we introduce two new families of multiset tree automata, by using previous results taken from tree language theory. We propose a formal definition of reversible multiset tree automata and local testable multiset tree automata. These features have been widely studied in previous works [6, 8].

The structure of this work is simple: first we give basic definitions and notation for tree languages, P systems and multiset tree automata and we define the new families of multiset tree automata. Finally, we give some guidelines for future research.

[⋆] Work supported by the Spanish Generalitat Valenciana under contract GV06/068.

2 Notation and definitions

In the sequel we will provide some concepts from formal language theory, membrane systems and multiset processing. We suggest the following books to the reader [12], [10] and [2].

Multisets

First, we will provide some definitions from multiset theory as exposed in [15].

Definition 2.1 *Let D be a set. A multiset over D is a pair $\langle D, f\rangle$ where $f : D \longrightarrow \mathbb{N}$ is a function.*

Definition 2.2 *Suppose that $A = \langle D, f\rangle$ and $B = \langle D, g\rangle$ are two multisets. The removal of multiset B from A, denoted by $A \ominus B$, is the multiset $C = \langle D, h\rangle$ where for all $a \in D$ $h(a) = max(f(a) - g(a), 0)$.*

Definition 2.3 *Let $A = \langle D, f\rangle$ be a multiset; we will say that A is* empty *if for all $a \in D$, $f(a) = 0$.*

Definition 2.4 *Let $A = \langle D, f\rangle$ and $B = \langle D, g\rangle$ be two multisets. Their sum, denoted by $A \oplus B$, is the multiset $C = \langle D, h\rangle$, where for all $a \in D$ $h(a) = f(a) + g(a)$.*

Definition 2.5 *Let $A = \langle D, f\rangle$ and $B = \langle D, g\rangle$ be two multisets. We will say that $A = B$ if the multiset $(A \ominus B) \oplus (B \ominus A)$ is empty.*

The size of any multiset M, denoted by $|M|$ will be the number of elements that it contains. We are specially interested in the class of multisets that we call *bounded multisets*. They are multisets that hold the property that the sum of all the elements is bounded by a constant n. Formally, we will denote by $\mathcal{M}_n(D)$ the set of all multisets $\langle D, f\rangle$ such that $\sum_{a \in D} f(a) = n$.

A concept that is quite useful to work with sets and multisets is the *Parikh mappings*. Formally, a Parikh mapping can be viewed as the application $\Psi : D^* \rightarrow \mathbb{N}^n$ where $D = \{d_1, d_2, \cdots, d_n\}$. Given an element $x \in D^*$ we define $\Psi(x) = (\#_{d_1}(x), \cdots, \#_{d_n}(x))$ where $\#_{d_j}(x)$ denotes the number of occurrences of d_j in x.

P systems

We will introduce basic concepts from membrane systems taken from [10]. A general P system of degree m is a construct

$$\Pi = (V, T, C, \mu, w_1, \cdots, w_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0),$$

where:

- V is an alphabet (the *objects*)
- $T \subseteq V$ (the *output alphabet*)

- $C \subseteq V$, $C \cap T = \emptyset$ (the *catalysts*)
- μ is a membrane structure consisting of m membranes
- w_i, $1 \leq i \leq m$ is a string representing a multiset over V associated with the region i
- R_i, $1 \leq i \leq m$ is a finite set of *evolution rules* over V associated with the ith region and ρ_i is a partial order relation over R_i specifying a *priority*.
  An evolution rule is a pair (u, v) (or $u \to v$) where u is a string over V and $v = v'$ or $v = v'\delta$ where v' is a string over
  $$\{a_{here}, a_{out}, a_{in_j} \mid a \in V, 1 \leq j \leq m\}$$
  and δ is an special symbol not in V (it defines the *membrane dissolving action*). From now on, we will denote the set *tar* by $\{here, out, in_k : 1 \leq k \leq m\}$.
- i_0 is a number between 1 and m and it specifies the *output* membrane of Π (in the case that it equals to ∞ the output is read outside the system).

The language generated by Π in external mode ($i_0 = \infty$) is denoted by $L(\Pi)$ and it is defined as the set of strings that can be defined by collecting the objects that leave the system by arranging the leaving order (if several objects leave the system at the same time then permutations are allowed). The set of numbers that represent the objects in the output membrane i_0 will be denote by $N(\Pi)$. Obviously, both sets $L(\Pi)$ and $N(\Pi)$ are defined only for *halting computations.*

One of the multiple variations of P systems is related to the creation, division and modification of membrane structures. There have been several works in which these variants have been proposed (see, for example, [1, 9, 10, 11]).

In the following, we enumerate some kind of rules which are able to modify the membrane structure:

1. 2-division: $[_h a]_h \to [_{h'} b]_{h'} [_{h''} c]_{h''}$
2. Creation: $a \to [_h b]_h$
3. Dissolving: $[_h a]_h \to b$

The power of P systems with the previous operations and other ones (e.g. *exocytosis, endocytosis*, etc.) has been widely studied in the previously related works and other ones.

Tree automata and tree languages

Now, we will introduce some concepts from tree languages and automata as exposed in [3, 5]. First, let a *ranked alphabet* be the association of an alphabet V together with a finite relation r in $V \times \mathbb{N}$. We denote by V_n the subset $\{\sigma \in V : (\sigma, n) \in r\}$.

The set V^T of trees over V, is defined inductively as follows:

$a \in V^T$ for every $a \in V_0$
$\sigma(t_1, ..., t_n) \in V^T$ whenever $\sigma \in V_n$ and $t_1, ..., t_n \in V^T$, $(n > 0)$

and let a *tree language* over V be defined as a subset of V^T.

Given the tuple $l =< 1, 2, ..., k >$ we will denote the set of permutations of l by $perm(l)$. Let $t = \sigma(t_1, ..., t_n)$ be a tree over V^T, we will denote the set of permutations of t at first level by $perm_1(t)$. Formally, $perm_1(t) = \{\sigma(t_{i_1}, ..., t_{i_n}) :< i_1, i_2, ..., i_n >\in perm(< 1, 2, ..., n >)\}$.

Let $\mathbb{N}^*$ be the set of finite strings of natural numbers, separated by dots, formed using the catenation as the composition rule and the empty word λ as the identity. Let the prefix relation $\leq$ in $\mathbb{N}^*$ be defined by the condition that $u \leq v$ if and only if $u \cdot w = v$ for some $w \in \mathbb{N}^*$ $(u, v \in \mathbb{N}^*)$. A finite subset D of $\mathbb{N}^*$ is called a *tree domain* if:

$$u \leq v \text{ where } v \in D \text{ implies } u \in D, \text{ and}$$
$$u \cdot i \in D \text{ whenever } u \cdot j \in D \; (1 \leq i \leq j)$$

Each tree domain D could be seen as an unlabeled tree whose nodes correspond to the elements of D where the hierarchy relation is the prefix order. Thus, each tree t over V can be seen as an application $t : D \to V$. The set D is called the *domain of the tree* t, and denoted by $dom(t)$. The elements of the tree domain $dom(t)$ are called *positions* or *nodes* of the tree t. We denote by $t(x)$ the label of a given node x in $dom(t)$.

Let the level of $x \in dom(t)$ be $|x|$. Intuitively, the level of a node measures its distance from the root of the tree. Then, we can define the depth of a tree t as $depth(t) = max\{|x| : x \in dom(t)\}$. In the same way, for any tree t, we denote the size of the tree by $|t|$ and the set of subtrees of t (denoted with $Sub(t)$) as follows:

$$Sub(a) = \{a\} \text{ for all } a \in V_0$$
$$Sub(t) = \{t\} \cup \bigcup_{i=1,...,n} Sub(t_i) \text{ for } t = \sigma(t_1, ..., t_n) \; (n > 0)$$

Given a tree $t = \sigma(t_1, \ldots, t_n)$, the root of t will be denoted as $root(t)$ and defined as $root(t) = \sigma$. If $t = a$ then $root(t) = a$. The successors of a tree $t = \sigma(t_1, \ldots, t_n)$ will be defined as $H^t =< root(t_1), \ldots, root(t_n) >$. Finally, $leaves(t)$ will denote the set of leaves of the tree t.

Definition 2.6 *A* finite deterministic tree automaton *is defined by the tuple* $A = (Q, V, \delta, F)$*: where* Q *is a finite set of states;* V *is a ranked alphabet,* $Q \cap V = \emptyset$*;* $F \subseteq Q$ *is the set of final states and* $\delta = \bigcup_{i:V_i \neq \emptyset} \delta_i$ *is a set of transitions defined as follows:*

$$\delta_n : (V_n \times (Q \cup V_0)^n) \to Q \qquad n = 1, \ldots, m$$
$$\delta_0(a) = a \qquad \forall a \in V_0$$

Given the state $q \in Q$, we define the *ancestors* of the state q, denoted by $Ant(q)$, as the set of strings

$$Ant(q) = \{p_1 \cdots p_n : p_i \in Q \cup V_0 \wedge \delta_n(\sigma, p_1, ..., p_n) = q\}$$

From now on, we will refer to finite deterministic tree automata simply as *tree automata*. We suggest [3, 5] for other definitions on tree automata.

The transition function δ is extended to a function $\delta : V^T \to Q \cup V_0$ on trees as follows:

$$\delta(a) = a \text{ for any } a \in V_0$$
$$\delta(t) = \delta_n(\sigma, \delta(t_1), \ldots, \delta(t_n)) \text{ for } t = \sigma(t_1, \ldots, t_n)\ (n > 0)$$

Note that the symbol δ denotes both the set of transition functions of the automaton and the extension of these functions to operate on trees. In addition, you can observe that the tree automaton A cannot accept any tree of depth zero.

Given a finite set of trees T, let the *subtree automaton* for T be defined as $AB_T = (Q, V, \delta, F)$, where:

$$\begin{aligned} &Q = Sub(T) \\ &F = T \\ &\delta_n(\sigma, u_1, \ldots, u_n) = \sigma(u_1, \ldots, u_n) \qquad && \sigma(u_1, \ldots, u_n) \in Q \\ &\delta_0(a) = a && a \in V_0 \end{aligned}$$

Let \$ be a new symbol in V_0, and $V_\$^T$ the set of trees $(V \cup \{\$\})^T$ where each tree contains \$ only once. We will name the node with label \$ as *link point* when necessary. Given $s \in V_\$^T$ and $t \in V^T$, the operation $s\#t$ is defined as:

$$s\#t(x) = \begin{cases} s(x) \text{ if } x \in dom(s),\ s(x) \neq \$ \\ t(z)\ \text{ if } x = yz,\ s(y) = \$,\ y \in dom(s) \end{cases}$$

therefore, given $t, s \in V^T$, let the tree quotient $(t^{-1}s)$ be defined as:

$$t^{-1}s = \begin{cases} r \in V_\$^T\ :\ s = r\#t \text{ if } t \in V^T - V_0. \\ t \qquad\qquad\qquad\quad \text{if } t \in V_0. \end{cases}$$

this quotient can be extended to consider set of trees $T \subseteq V^T$ as:

$$t^{-1}T = \{t^{-1}s\ :\ s \in T\}$$

For any $k \geq 0$, let the k-root of a tree t be defined as follows:

$$root_k(t) = \begin{cases} t, & \text{if } depth(t) < k \\ t' : t'(x) = t(x),\ x \in dom(t) \wedge |x| \leq k, & \text{otherwise} \end{cases}$$

Multiset tree automata and mirrored trees

We will extend over multisets some definitions of tree automata and tree languages. We will introduce the concept of multiset tree automata and then we will characterize the set of trees that it accepts.

Given any tree automata $A = (Q, V, \delta, F)$ and $\delta_n(\sigma, p_1, p_2, \ldots, p_n) \in \delta$, we can associate to δ_n the multiset $\langle Q \cup V_0, f\rangle \in \mathcal{M}_n(Q \cup V_0)$ where f is defined by $\Psi(p_1 p_2 \ldots p_n)$. The multiset defined in such way will be denoted by $M_\Psi(\delta_n)$. Alternatively, we can define $M_\Psi(\delta_n)$ as $M_\Psi(p_1) \oplus M_\Psi(p_2) \oplus \cdots \oplus M_\Psi(p_n)$ where $\forall 1 \leq i \leq n$ $M_\Psi(p_i) \in \mathcal{M}_1(Q \cup V_0)$. Observe that if $\delta_n(\sigma, p_1, p_2, \ldots, p_n) \in \delta$, $\delta'_n(\sigma, p'_1, p'_2, \ldots, p'_n) \in \delta$ and $M_\Psi(\delta_n) = M_\Psi(\delta'_n)$ then δ_n and δ'_n are defined over the same set of states and symbols but in different order (that is the multiset induced by $\langle p_1, p_2, \cdots, p_n\rangle$ equals to the one induced by $\langle p'_1 p'_2 \ldots p'_n\rangle$).

Now, we can define a *multiset tree automaton* that performs a bottom-up parsing as in the tree automaton case.

Definition 2.7 *A* multiset tree automaton *is defined by the tuple $MA = (Q, V, \delta, F)$, where Q is a finite set of states, V is a ranked alphabet with $maxarity(V) = n$, $Q \cap V = \emptyset$, $F \subseteq Q$ is a set of final states and δ is a set of transitions defined as follows:*

$$\delta = \bigcup_{\substack{1 \leq i \leq n \\ i : V_i \neq \emptyset}} \delta_i$$

$$\begin{aligned} \delta_i : (V_i \times \mathcal{M}_i(Q \cup V_0)) &\to \mathcal{P}(\mathcal{M}_1(Q)) \qquad && i = 1, \ldots, n \\ \delta_0(a) = M_\Psi(a) &\in \mathcal{M}_1(Q \cup V_0) \qquad && \forall a \in V_0 \end{aligned}$$

We can take notice that every tree automaton A defines a multiset tree automaton MA as follows

Definition 2.8 *Let $A = (Q, V, \delta, F)$ be a tree automaton. The multiset tree automaton induced by A is defined by the tuple $MA = (Q, V, \delta', F)$ where each δ' is defined as follows: $M_\Psi(r) \in \delta'_n(\sigma, M)$ if $\delta_n(\sigma, p_1, p_2, ..., p_n) = r$ and $M_\Psi(\delta_n) = M$.*

Observe that, in the general case, the multiset tree automaton induced by A is non deterministic.

As in the case of tree automata, δ' could also be extended to operate on trees. Here, the automaton carries out a bottom-up parsing where the tuples of states and/or symbols are transformed by using the Parikh mapping Ψ to obtain the multisets in $\mathcal{M}_n(Q \cup V_0)$. If the analysis is completed and δ' returns a multiset with at least one final state, the input tree is accepted. So, δ' can be extended as follows

$\delta'(a) = M_\Psi(a)$ for any $a \in V_0$
$\delta'(t) = \{M \in \delta'_n(\sigma, M_1 \oplus \cdots \oplus M_n) : M_i \in \delta'(t_i) 1 \leq i \leq n\}$ for $t = \sigma(t_1, \ldots, t_n)$ $(n > 0)$

Formally, every multiset tree automaton MA accepts the following language

$$L(MA) = \{t \in V^T : M_\Psi(q) \in \delta'(t), q \in F\}$$

Another extension which will be useful is the one related to the ancestors of every state. So, we define $Ant_\Psi(q) = \{M : M_\Psi(q) \in \delta_n(\sigma, M)\}$.

Theorem 2.9 *(Sempere and López, [13]) Let $A = (Q, V, \delta, F)$ be a tree automaton, $MA = (Q, V, \delta', F)$ be the multiset tree automaton induced by A and $t = \sigma(t_1, \ldots, t_n) \in V^T$. If $\delta(t) = q$ then $M_\Psi(q) \in \delta'(t)$.*

Corolary 2.10 *(Sempere and López, [13]) Let $A = (Q, V, \delta, F)$ be a tree automaton and $MA = (Q, V, \delta', F)$ be the multiset tree automaton induced by A. If $t \in L(A)$ then $t \in L(MA)$.*

We will introduce the concept of *mirroring* in tree structures as exposed in [13]. Informally speaking, two trees will be related by mirroring if some permutations at the structural level are hold. We propose a definition that relates all the trees with this mirroring property.

Definition 2.11 *Let t and s be two trees from V^T. We will say that t and s are* mirror equivalent*, denoted by $t \bowtie s$, if one of the following conditions holds:*

1. $t = s = a \in V_0$
2. $t \in perm_1(s)$
3. $t = \sigma(t_1, \ldots, t_n)$, $s = \sigma(s_1, \ldots, s_n)$ *and there exists* $< s^1, s^2, \ldots, s^k > \in perm(< s_1, s_2, ..., s_n >)$ *such that* $\forall 1 \leq i \leq n$ $t_i \bowtie s^i$

Theorem 2.12 *(Sempere and López, [13]) Let $A = (Q, V, \delta, F)$ be a tree automaton, $t = \sigma(t_1, \ldots, t_n) \in V^T$ and $s = \sigma(s_1, \ldots, s_n) \in V^T$. Let $MA = (Q, V, \delta', F)$ be the multiset tree automaton induced by A. If $t \bowtie s$ then $\delta'(t) = \delta'(s)$.*

Corolary 2.13 *(Sempere and López, [13]) Let $A = (Q, V, \delta, F)$ be a tree automaton, $MA = (Q, V, \delta', F)$ the multiset tree automaton induced by A and $t \in V^T$. If $t \in L(MA)$ then, for any $s \in V^T$ such that $t \bowtie s$, $s \in L(MA)$.*

The last results were useful to propose an algorithm to determine whether two trees are mirror equivalent or not [13]. So, given two trees s and t, we can establish in time $\mathcal{O}((min\{|t|, |s|\})^2)$ if $t \bowtie s$.

3 *k*-testable in the strict sense (k-TSS) multiset tree languages and reversible multiset tree languages

In the following section, we will define two new classes of multiset tree languages. The definitions related to multiset tree automata come from the relation between mirrored trees and multiset tree automata which we have established in the previous section. So, whenever we refer to multiset tree languages we are taking under our consideration the set of (mirrored) trees accepted by multiset tree automata.

We refer [6] in order to know more about reversibility and local testability in tree languages.

First, we define k-TSS tree languages for any $k \geq 2$.

Definition 3.1 *Let $T \subseteq V^T$ and the integer value $k \geq 2$. T is a k-TSS multiset tree language if and only if, given whatever two trees $u_1, u_2 \in V^T$ such that $root_{k-1}(u_1) = root_{k-1}(u_2)$, $u_1^{-1}T \neq \emptyset$ and $u_2^{-1}T \neq \emptyset$ implies that $u_1^{-1}T = u_2^{-1}T$*

Any multiset tree automaton that holds the definition given before will be named a k-TSS multiset tree automaton. We can give the following characterization of such automata.

Corolary 3.2 *Let A be a k-TSS multiset tree automaton. There not exist two distinct states q_1, q_2 such that $root_k(q_1) \cap root_k(q_2) \neq \emptyset$*

Example 3.3 *Consider the multiset tree automaton with transitions:*

$$\begin{array}{r@{\,=\,}l}
\hline
\delta(\sigma, aa) & q_1 \\
\delta(\sigma, a) & q_2 \\
\delta(\sigma, aq_2) & q_2 \\
\delta(\sigma, q_1q_1) & q_1 \\
\hline
\delta(\sigma, aq_2q_1) & q_3 \in F \\
\hline
\end{array}$$

Note that the multiset tree language accepted by the automaton is k-TSS for any $k \geq 2$.

Note also that the following one does not hold the k-TSS condition for any $k \geq 2$:

$$\begin{array}{r@{\,=\,}l}
\hline
\delta(\sigma, aa) & q_1 \\
\delta(\sigma, bb) & q_2 \\
\delta(\sigma, q_2q_2) & q_2 \\
\delta(\sigma, q_1q_1) & q_1 \\
\hline
\delta(\sigma, q_2q_1) & q_3 \in F \\
\hline
\end{array}$$

because both the states q_1 and q_2 (and q_3) share a common k-root.

□

We also extend a previous result concerning k-reversible tree languages (for any $k \geq 0$) to give the following definition.

Definition 3.4 *Let $T \subseteq V^T$ and the integer value $k \geq 0$. T is a k-reversible multiset tree language if and only if, given whatever two trees $u_1, u_2 \in V^T$ such that $root_{k-1}(u_1) = root_{k-1}(u_2)$, whenever there exists a context $t \in V_\$^T$ such that both $u_1\#t, u_1\#t \in T$, then $u_1^{-1}T = u_2^{-1}T$*

Example 3.5 *Consider the multiset tree automaton with transitions:*

$$\begin{array}{r}
\hline
\delta(\sigma, aa) = q_1 \\
\delta(\sigma, a) = q_2 \\
\delta(\sigma, q_2q_2) = q_2 \\
\delta(\sigma, aaq_1) = q_1 \\
\hline
\delta(\sigma, q_1q_1) = q_3 \in F \\
\hline
\delta(\sigma, q_2q_1) = q_3 \in F \\
\hline
\end{array}$$

the multiset tree language accepted by this automaton is k-reversible and it is also an example of non k-TSS multiset tree language.

□

Finally, we can relate the two families of multiset tree languages that we have previously defined with the following result.

Theorem 3.6 *Let $T \subseteq V^T$ and an integer value $k \geq 2$, if T is k-TSS then T is $(k-1)$-reversible.*

Proof.

Let $t\#t_1$ and $t\#t_2$ belong to T, with $t \in V_\$^T$ and $root_k(t_1) = root_k(t_2)$, trivially $t_1^{-1}T \neq \emptyset$ and $t_2^{-1}T \neq \emptyset$. If T is a k-TSS tree language, then by previous definitions, $t_1^{-1}T = t_2^{-1}T$, and also T is $(k-1)$-reversible. □

4 Conclusions and future work

We have introduced two new families of multiset tree languages. Now, the open question is the characterization of membrane structures defined by them. We think that reversibility and local testability will introduce restrictions in the way of defining membrane creation and deletion. This will be explored in future works.

References

1. A. Alhazov, T.O. Ishdorj: Membrane operations in P systems with active membranes. In *Proc. Second Brainstorming Week on Membrane Computing.* TR 01/04 of RGNC, Sevilla University, 2004, 37–44.
2. C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Multiset Processing.* LNCS 2235, Springer, 2001.

3. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi: *Tree automata techniques and applications.* Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997, release October, 1rst 2002.
4. R. Freund, M. Oswald, A. Păun: P systems generating trees. In *Pre-proceedings of Fifth Workshop on Membrane Computing*, WMC5 (G. Mauri, Gh. Păun, C. Zandron, eds.), MolCoNet project IST-2001-32008, 2004, 221–232.
5. F. Gécseg, M. Steinby: Tree languages. In *Handbook of Formal Languages*, volume 3, Springer, Berlin, 1997, 1–69.
6. D. López: *Inferencia de lenguajes de árboles.* PhD Thesis DSIC, Universidad Politécnica de Valencia, 2003.
7. D. López, J.M. Sempere: Editing distances between membrane structures. In *Proceedings of the 6th International Workshop on Membrane Computing*, Vienna, 2005 (R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), LNCS 3850, Springer, 2006, 326–341.
8. D. López, J. M. Sempere, P. García: nference of reversible tree languages. *IEEE Transactions on Systems, Man and Cybernetics*, Part B: *Cybernetics*, 34, 4 (2004), 1658–1665.
9. A. Păun: On P systems with active membranes. In *Proc. of the First Conference on Unconventionals Models of Computation* (UMC2K), 2000, 187–201.
10. Gh. Păun: *Membrane Computing. An Introduction.* Springer, 2002.
11. Gh. Păun, Y. Suzuki, H. Tanaka, T. Yokomori: On the power of membrane division on P systems. In *Proc. Conf. on Words, Languages and Combinatorics*, Kyoto, 2000.
12. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages.* Springer, 1997.
13. J.M. Sempere, D. López :Recognizing membrane structures with tree automata. In *3rd Brainstorming Week on Membrane Computing, Sevilla, 2005.* RGNC Report 01/2005 Research Group on Natural Computing, Sevilla University, Fenix Editora, Sevilla, 2005, 305–316.
14. J.M. Sempere, D. López: Identifying P rules from membrane structures with an error-correcting approach. In *Proceedings of the 7th International Workshop on Membrane Computing*, Leiden, 2006 (H.J. Hoogeboom, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), LNCS 4361, Springer, 2006, 507–520.
15. A. Syropoulos: Mathematics of multisets. In [2], 347–358.