# Characterizing Membrane Structures Through Multiset Tree Automata

José M. Sempere and Damián López

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n 46017 Valencia, Spain
{jsempere,dlopez}@dsic.upv.es

**Abstract.** 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. For instance, in [13], the authors propose an extension of tree automata, namely multiset tree automata, in order to recognize membrane structures. In [7], this model is used to calculate editing distances between membrane structures. Later, a method to infer multiset tree automata from membrane observations was presented in [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 as follows: 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.

## 2 Notation and Definitions

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

G. Eleftherakis et al. (Eds.): WMC8 2007, LNCS 4860, pp. 428–437, 2007.

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

**Definition 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. We say that $A$ is* empty *if for all $a \in D$, $f(a) = 0$.*

**Definition 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 $a \in D$ $h(a) = max(f(a) - g(a), 0)$. Their sum, denoted by $A \oplus B$, is the multiset $C = \langle D, h\rangle$, where for all $a \in D$ $h(a) = f(a) + g(a)$.*

*Then, we 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 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 mapping*. 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 $x$.

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

$$\Pi = (V, T, C, \mu, w_1, \cdots, w_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0), \text{ where:}$$

- $V$ is an alphabet (the *objects*)
- $T \subseteq V$ (the *output alphabet*)
- $C \subseteq V$, $C \cap T = \emptyset$ (the *catalysts*)
- $\mu$ 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 $i$th region and $\rho_i$ is a partial order relation over $R_i$ specifying a *priority*.
  An evolution rule is a pair $(u, v)$ (or $u \rightarrow v$) 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 $\delta$ is a 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 $\Pi$ (in the case that it equals to $\infty$ the output is read outside the system).

The language generated by $\Pi$ 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 them in the leaving order (if several objects leave the system at the same time then permutations are allowed). The set of

vector numbers that represent the objects in the output membrane $i_0$ will be denoted 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: $[_h a]_h \rightarrow [_{h'} b]_{h'} [_{h''} c]_{h''}$
2. Creation: $a \rightarrow [_h b]_h$
3. Dissolving: $[_h a]_h \rightarrow b$

The power of P systems with the previous operations and other ones (*exocytosis, endocytosis*, etc.) has been widely studied in the membrane computing area.

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 $\{\sigma \in V \mid (\sigma, n) \in r\}$.

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 $l = \langle 1, 2, ..., k \rangle$ 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 denote the set of permutations of $t$ at first level by $perm_1(t)$. Formally, $perm_1(t) = \{\sigma(t_{i_1}, ..., t_{i_n}) \mid \langle i_1, i_2, ..., i_n \rangle \in perm(\langle 1, 2, ..., n \rangle)\}$.

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 $\lambda$ 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 $t : D \rightarrow V$. 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 $H^t = \langle root(t_1), \ldots, root(t_n)\rangle$. Finally, $leaves(t)$ will denote the set of leaves of the tree $t$.

**Definition 3.** *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 with* $m$ *as the maximum integer in the relation* $r$, $Q \cap V = \emptyset$; $F \subseteq Q$ *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

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

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 $\delta$ 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 $\delta$ 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:

$$Q = Sub(T)$$
$$F = T$$
$$\delta_n(\sigma, u_1, \ldots, u_n) = \sigma(u_1, \ldots, u_n) \qquad \sigma(u_1, \ldots, u_n) \in Q$$
$$\delta_0(a) = a \qquad a \in V_0$$

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

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

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

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

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

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

## 3 Multiset Tree Automata and Mirrored Trees

We extend now 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 automaton $A = (Q, V, \delta, F)$ and $\delta_n(\sigma, p_1, p_2, \ldots, p_n) \in \delta$, we can associate to $\delta_n$ 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 $\delta_n$ and $\delta'_n$ 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 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 4.** *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* $\delta$ *is a set of transitions defined as follows:*

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

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

We can observe that every tree automaton $A$ defines a multiset tree automaton $MA$ as follows

**Definition 5.** *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 $\delta'$ is defined as follows: $M_\Psi(r) \in \delta'_n(\sigma, M)$ if $\delta_n(\sigma, p_1, ..., p_n) = r$ 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, $\delta'$ 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 $\Psi$ to obtain the multisets in $\mathcal{M}_n(Q \cup V_0)$. If the analysis is completed and $\delta'$ returns a multiset with at least one final state, the input tree is accepted. So, $\delta'$ can be extended as follows

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

Formally, every multiset tree automaton $MA$ accepts the following language

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

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

**Theorem 1.** *(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 $M_\Psi(q) \in \delta'(t)$.*

**Corollary 1.** *(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 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 hold. We propose a definition that relates all the trees with this mirroring property.

**Definition 6.** *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* $\langle s^1, s^2, \ldots, s^k \rangle \in perm(\langle s_1, s_2, ..., s_n \rangle)$ *such that* $\forall 1 \leq i \leq n\ t_i \bowtie s^i$

**Theorem 2.** *(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 $\delta'(t) = \delta'(s)$.*

**Corollary 2.** *(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 $t \bowtie s$.

## 4 *k*-Testable in the Strict Sense (k-TSS) Multiset Tree Languages

In this section, we will define a new class 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 to [6] for more details about reversibility and local testability in tree languages.

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

**Definition 7.** *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 as in the definition given before will be named a $k$-TSS multiset tree automaton. Given $A$ a tree automaton, $t_1$, $t_2$ valid contexts over $V_{\$}^T$, as an extension of a result concerning $k$-TSS tree languages, we give the following definition:

**Definition 8.** *Let $A$ be a multiset tree automaton over $V_{\$}^T$. Let $u_1, u_2 \in V^T$ be two trees such that $root_{k-1}(u_1) = root_{k-1}(u_2)$ and $t_1 \# u_1, t_2 \# u_2 \in L(A)$ for some valid contexts $t_1$ and $t_2$. If $A$ is a $k$-TSS mirror tree automaton then $\delta(u_1) = \delta(u_2)$.*

We can give the following characterization of such automata.

**Corollary 3.** *Let $A$ be a $k$-TSS multiset tree automaton. There does not exist two distinct states $q_1, q_2$ such that $root_k(q_1) \cap root_k(q_2) \neq \emptyset$.*

The previous result can be easily deduced from the definition of $k$-TSS multiset tree automata and the definitions given in section 2 about tree automata and tree languages.

*Example 1.* Consider the multiset tree automaton with transitions:

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

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 have the $k$-TSS condition for any $k \geq 2$:

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

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

## 5 Reversible Multiset Tree Automata

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

**Definition 9.** *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 $u_1\#t, u_2\#t \in T$, then $u_1^{-1}T = u_2^{-1}T$.*

**Definition 10.** *Let $A$ be a multiset tree automaton over $V_{\$}^T$. Let $p_1, p_2 \in Q$ be two states such that $root_k(L(p_1)) \cap root_k(L(p_2)) \neq \emptyset$. $A$ is order $k$ reset free if the automaton does not contain two transitions such that*

$$\delta(\sigma, q_1q_2 \ldots q_np_1) = \delta(\sigma, q_1q_2 \ldots q_np_2)$$

*where $q_i \in Q,\ 1 \leq i \leq n$.*

**Definition 11.** *Let $A$ be a multiset tree automaton. $A$ is $k$-reversible if $A$ is order $k$ reset free and for any two distinct final states $f_1$ and $f_2$ the condition $root_k(L(f_1)) \cap root_k(L(f_2)) = \emptyset$ is fulfilled.*

*Example 2.* Consider the multiset tree automaton with transitions:

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

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.** *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. □

## 6 From Transitions to Membrane Structures

Once we have formally defined the two classes of multiset tree automata, we will translate their characteristics in terms of membrane structures. First we will give a meaning to the concept of $root_k(t)$. Observe that in a membrane structure $t$, which is represented by a set of mirrored trees $\{t' \mid t \bowtie t'\}$, the meaning of $root_k(t)$ is established by taking into account the (sub)structure of the membranes from the top region up to a depth of length $k$. Another concept that we have managed before is the operator #. Observe that this is related to membrane creation of P systems. So, we can go from membrane configuration $t$ to $t\#s$ by creating a new membrane structure $s$ in a predefined region of $t$ (established by #).

So, $k$ testability implies that, whenever we take two membrane (sub)structures of the P system, $u_1$ and $u_2$, if they share a common substructure of order $k-1$ then $u_1$ appears in a membrane configuration if and only if it can be substituted by $u_2$ to give a different membrane configuration.

On the other hand, $k$-reversibility implies that whenever two membrane structures $u_1$ and $u_2$ share the same substructure up to length $k-1$, if $u_1\#t$ and $u_2\#t$ are substructures of valid configurations of the P system, then $u_1\#s$ is a substructure of a valid configuration of the P system if and only if so is $u_2\#s$.

## 7 Conclusions and Future Work

We have introduced two new families of multiset tree languages. These classes have characterized the membrane structures defined by P systems. We think that other classes of tree languages will imply new classes of membrane structures. So, all the theory that has been previously established on tree languages can enrich the way in which we look up to the membrane structures.

In addition, there is another way to explore the relation between the membrane structures of P systems and the languages that they can accept or generate. So, a natural question arises: How is affected the structure of the language by the structure of the membranes? This issue will be explored in future works.

## Acknowledgements

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

## References


1. Alhazov, A., Ishdorj, T.O.: Membrane operations in P systems with active membranes. In: Proc. Second Brainstorming Week on Membrane Computing. TR 01/04 of RGNC, pp. 37–44, Sevilla University (2004)
2. Calude, C.S., Păun, G., Rozenberg, G., Salomaa, A. (eds.): Multiset Processing. LNCS, vol. 2235. Springer, Heidelberg (2001)
3. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (1997) (release October 1, 2002), available on: `http://www.grappa.univ-lille3.fr/tata`
4. Freund, R., Oswald, M., Păun, A.: P Systems Generating Trees. In: Mauri, G., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 309–319. Springer, Heidelberg (2005)
5. Gécseg, F., Steinby, M.: Handbook of Formal Languages, ch. Tree languages, vol. 3, pp. 1–69. Springer, Heidelberg (1997)
6. López, D.: Inferencia de lenguajes de árboles. PhD Thesis DSIC, Universidad Politécnica de Valencia (2003)
7. López, D., Sempere, J.M.: Editing Distances between Membrane Structures. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 326–341. Springer, Heidelberg (2006)
8. López, D., Sempere, J.M., García, P.: Inference of reversible tree languages. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics 34(4), 1658–1665 (2004)
9. Păun, A.: On P systems with active membranes. In: UMC 2000. Proc. of the First Conference on Unconventionals Models of Computation, pp. 187–201 (2000)
10. Păun, G.: Membrane Computing. An Introduction. Springer, Heidelberg (2002)
11. Păun, G., Suzuki, Y., Tanaka, H., Yokomori, T.: On the power of membrane division on P systems. Theoretical Computer Science 324(1), 61–85 (2004)
12. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 1. Springer, Heidelberg (1997)
13. Sempere, J.M., López, D.: Recognizing membrane structures with tree automata. In: Gutiérrez Naranjo, M.A., Riscos-Núñez, A., Romero-Campero, F.J., Sburlan, D. (eds.) 3rd Brainstorming Week on Membrane Computing 2005. RGNC Report 01/2005 Research Group on Natural Computing, Sevilla University, pp. 305–316. Fénix Editora (2005)
14. Sempere, J.M., López, D.: Identifying P Rules from Membrane Structures with an Error-Correcting Approach. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2006. LNCS, vol. 4361, pp. 507–520. Springer, Heidelberg (2006)
15. Syropoulos, A.: Mathematics of Multisets. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 347–358. Springer, Heidelberg (2001)