Recognizing Membrane Structures with Tree Automata*

José M. Sempere, Damián López

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46071 Valencia, Spain
E-mail: {jsempere,dlopez}@dsic.upv.es

Summary. In this work we propose a new model of tree automata based on multisets of states and symbols linked to the finite control. This new model accepts a set of trees with symmetries between their internal nodes. We name this property as *mirroring*. We propose an application of these automata to solve a problem related to P systems such as recognizing identic membrane structures.

1 Introduction

One of the main components of P systems is the membrane structure that they define. This structure evolves during the computation time due to the application of rules associated to the membranes. The membrane structure can be represented by a tree in which the internal nodes denote regions which have inner regions inside. So, the root of the tree is always associated to the skin membrane of the P system.

The relation between regions and trees has been recently stressed by Freund *et al.* [6]. These authors have established that any recursively enumerable set of trees can be generated by a P system with active membranes and string objects. So, P systems can be viewed as tree generators.

In this work we propose a model to accept the tree structures defined by P systems. Our model is an extension of classical tree automata [7] in which the states and symbols of the finite control form *multisets*. Multiset theory has been linked to parallel processing as shown in [2].

The use of multisets in the finite control of tree automata implies the definition of a multiset tree automaton model. This new model is able to process a set of trees which have some symmetry properties linked to the internal nodes. These symmetries appear during the representation of membrane structures of P systems. That is, the initial order given to the regions of any P system is somehow arbitrary.

* Work supported by the Spanish CICYT under contract TIC2003-09319-C03-02.

The renaming of regions and rules in any P system does not change its behavior during the computation. So, we have an exponential number of names that we can give to any P system. This fact is what we try to study by introducing symmetries (*mirrorings*) in the membrane structure.

The problem of representation that we have referred before opens a new definition of *confluent* P systems. We can define *structural confluent* P systems as those systems in which the uniqueness of acceptance/rejection configurations is only referred to the membrane structure that they define. This will be investigated in future works.

Another aspect that we try to solve is related to *editing configurations*. Recently, Csuhaj-Varjú *et al.* [4] have proposed editing distances between configurations of P systems. Here, we initiate the first step towards editing structural configuration of P systems. That is, we can restrict the editing distances by taking into account only membrane structures. The new model that we propose in this work will be useful to calculate such distances. Here we can take advantage of a previous work on editing distances between trees and tree automata [9].

The structure of this work is as follows. First we introduce basic definitions and notation about multisets, tree languages and automata and P systems. In section 3, we define the new model of multiset tree automata. We define the relation of *mirroring* between trees and we establish some results between tree automata, multiset tree automata and mirroring trees. In section 4, we apply the new model to solve the acceptance of membrane structures and the equivalence between them. Finally, we draw some conclusions and give some guidelines for future works.

2 Notation and Definitions

In the sequel we will provide some concepts from formal language theory, membrane systems and multiset processing. We suggest the books [14], [11], and [2] as basic references.

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.*

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 $\Psi : D^* \to \mathbb{N}^n$ 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.

Later, we will use tuples of symbols and states as strings and we will apply the Parikh mapping as defined above.

Tree automata and tree languages

Now, we introduce some concepts from tree languages and automata as exposed in [3, 7]. 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 $\{\sigma \in V \mid (\sigma, n) \in r\}$.

Given a ranked alphabet V with r in $V \times \mathbb{N}$, we define the maximum arity of the alphabet as $maxarity(V) = max\{n \mid (\sigma, n) \in r\}$.

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

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

A *tree language* over V is defined as a subset of V^T.

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

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 unlabelled 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 $depth(t) = max\{|x| \mid x \in dom(t)\}$. 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:

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

Definition 2.6 *A* finite deterministic tree automaton *is defined as a system* $A = (Q, V, \delta, F)$*, 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:*

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

From now on, we will refer to finite deterministic tree automata simply as *tree automata*. We suggest [3, 7] 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, one 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:

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

P systems

Finally, we will introduce basic concepts from membrane systems taken from [11]. 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 $\{here, out, in_k \mid 1 \leq k \leq m\}$ by tar.
- 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 them in 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, 10, 11, 12]).

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.

3 Multiset Tree Automata and Mirrored Trees

We will extend over multisets some definitions of tree automata and tree languages. First, we will introduce the concept of multiset tree automata and then we will try to characterize the set of trees that it accepts. Observe that our approach is a

little bit different from Csuhaj-Varjú *et al.* [5] and from Kudlek *et al.* [8] where, the authors consider the case that *bags of objects* are analyzed by an abstract machine. Here, we do not consider *bags of (sub)trees* but we introduce bags of states and symbols in the finite control of the automata.

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 this 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 for all $1 \leq i \leq n$, $M_\Psi(p_i) \in \mathcal{M}_1(Q \cup V_0)$. 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 3.1 *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:*

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

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

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

Definition 3.2 *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* $\delta_n(\sigma, p_1, p_2, \ldots, p_n) = r$ *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 analyzes the leaves of the tree first through δ_0. Then, by making a bottom-up parsing, the tuples of states and/or symbols are transformed in multisets 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

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

Formally, every multiset tree automaton MA accepts the following language

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

Theorem 3.1 *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 $M_\Psi(q) \in \delta'(t)$.*

Proof. We will proceed by induction on the depth of the tree.
Base of induction:

Let us consider that $t = \sigma(a_1, \ldots, a_n)$ with $a_i \in V_0$ and $\delta_n(\sigma, a_1, \cdots, a_n) = q$. The definition of δ' implies that $M_\Psi(q) \in \delta'(\sigma, M_\Psi(a_1 a_2 \ldots a_n))$ so $M_\Psi(q) \in \delta'(t)$.
Induction hypothesis:

Let us consider that $depth(t) < n$. If $\delta(t) = q$ then $M_\Psi(q) \in \delta'(t)$.
Induction step:

$depth(t) = n$. Let us suppose that $t = \sigma(t_1, \cdots, t_n)$ with $depth(t_i) < n$ for $1 \leq i \leq n$. Our induction hypothesis establishes that, if $\delta(t_i) = q_i$, then $M_\Psi(q_i) \in \delta'(t_i)$ for $1 \leq i \leq n$, Then, $\delta'_n(\sigma, t_1, \cdots, t_n) = \delta'_n(\sigma, P)$ with $P = M_\Psi(q_1 \ldots q_n) \in \mathcal{M}_n(Q \cup V_0)$. So, if $\delta_n(\sigma, q_1, \cdots, q_n) = r$, then the construction of δ' ensures that $M_\Psi(r) \in \delta'_n(\sigma, P) = \delta'(t)$.

Corollary 3.1 *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 $t \in L(MA)$.*

Mirrored equivalent trees

We now introduce the concept of *mirroring* in tree structures. Informally speaking, two trees will be related by mirroring if some permutations at the structural level are mad. For example, the trees given in Figure 1 have identical subtrees except that some internal nodes have changed their order.

We propose a definition that relates all the trees with this mirroring property.

Definition 3.3 *Let t and s be two trees from V^T. We 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* $< s^1, s^2, \ldots, s^k > \in perm(< s_1, s_2, \ldots, s_n >)$ *such that* $t_i \bowtie s^i$ *for all* $1 \leq i \leq n$.

Property 3.1 *$\bowtie$ is a binary equivalence relation (i.e., it is reflexive, symmetric and transitive).*

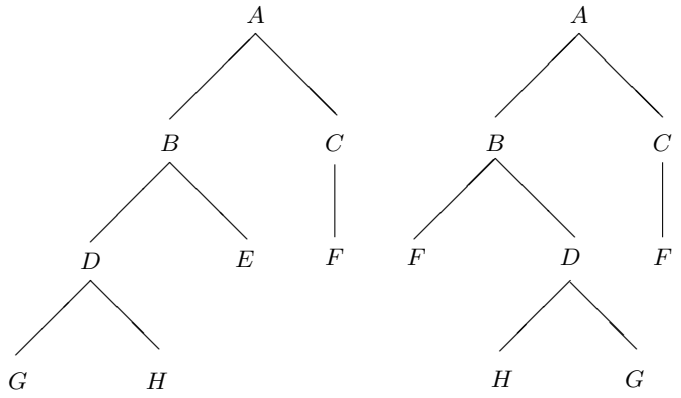

Fig. 1. Two mirrored trees.

Theorem 3.2 *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 $\delta'(t) = \delta'(s)$.*

Proof. We will proceed by induction on the depth of the tree.
Base of induction:

Let us take t and s with $depth(t) = depth(s) = 1$. It is easy to see that the transition function δ' consider in the same way every permutation of the leaves of t, that is, every mirror equivalent tree to t.
Induction hypothesis:

Let us consider that $depth(t) = depth(s) < n$. If $t \bowtie s$, then $\delta'(t) = \delta'(s)$.
Induction step:

Finally, take t and s with $depth(t) = depth(s) = n$. Let us suppose that $t \bowtie s$ with $t = \sigma(t_1, \cdots, t_n)$ and $s = \sigma(s_1, \cdots, s_n)$. Therefore, for each t_i there exist s_j such that $t_i \bowtie s_j$, $1 \leq i, j \leq n$ and, due to our induction hypothesis, $\delta'(t_i) = \delta'(s_j)$. So, the set of multisets defined by $\delta'(t_1) \oplus \cdots \oplus \delta'(t_n)$ is equal to the set of multisets defined by $\delta'(s_1) \oplus \cdots \oplus \delta'(s_n)$ and $\delta'(t) = \delta'(s)$.

Note that the reciprocal of last theorem is not generally true. For instance, consider the trees $t = \sigma(a)$ and $s = \sigma(a, \sigma(a))$ and the tree automaton with the following transition function:

$$\delta_1(\sigma, a) = q_1 \in F, \qquad \delta_2(\sigma, a, q_1) = q_1 \in F.$$

It is easy to see that $\delta'(t) = \delta'(s)$ but t is not mirror equivalent to s.

Corollary 3.2 *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 $s \in L(MA)$ for any $s \in V^T$ such that $t \bowtie s$.*

Proof. Note that if $t \in L(A)$ then $t \in L(MA)$ due to Theorem 3.1 and Corollary 3.1. Finally, by Theorem 3.2, if $t \bowtie s$, then $\delta'(t) = \delta'(s)$ and $s \in L(MA)$.

This result suggests an algorithm to determine whether two trees are mirror equivalent or not. The algorithm is shown in Figure 2

Input:
Two trees $t, s \in V^T$
Output:
Answer to $t \bowtie s$
Method:
1. Let $AB_t = (Q, V, \delta, F)$ be the subtree automaton for the set $\{t\}$
2. Let $MAB_t = (Q, V, \delta', F)$ be the mirrored tree automaton induced by AB_t
3. **if** $(\delta'(s) \cap F \neq \emptyset)$
   then return *TRUE*
   else return *FALSE*
   fi

Fig. 2. Algorithm to determine if two trees are mirror equivalent.

Note that given $s, t \in V^T$, the algorithm constructs AB_t and MAB_t. By Corollary 3.1, $t \in L(MAB_t)$ and, due to Corollary 3.2, if $t \bowtie s$, then $s \in L(MAB_t)$. In addition, if t is not mirror equivalent to s, then $\delta'(s)$ is not defined due to the construction of AB_t which is deterministic and accepts only t.

Observe that the multiset tree automaton MAB_t could be non-deterministic. For example, if $\delta(\sigma, a, b) = q_1$ and $\delta(\sigma, b, a) = q_2$ is defined for AB_t, then $\delta'(\sigma, M_\Psi(ab)) = \{M_\Psi(q_1), M_\Psi(q_2)\}$ is defined for MAB_t. This case could be avoided: just rename the state q_2 as q_1 during the construction in step 2 of the proposed algorithm. So, we can assume that MAB_t is deterministic too. The subtree automaton AB_t proposed in the algorithm contains neither loops nor cycles.

It is possible to establish in polynomial time if MAB_t accepts a given tree s. Furthermore, we will prove that the proposed algorithm runs in polynomial time with the size of the input tree t.

Lemma 3.1 *Algorithm of Figure 2 runs in time* $\mathcal{O}(|t|^2)$.

Proof. First, the operations $\oplus$ and the symmetric difference, defined over multisets, can be performed in time $\mathcal{O}(n)$ where n is the size of the set that defines the multiset [13]. In this case, $n = |t|$.

Now we will analyze the time complexity of the algorithm step by step

Step 1 runs in time $\mathcal{O}(|t|)$.

Step 2 runs in time $\mathcal{O}(|t|^2)$. Observe that the number of multiset operations involved in the construction of MAB_t is bounded by the size of the automaton AB_t which is bounded by $|t|$.

Step 3 runs in time $\mathcal{O}(|t|^2)$. Here, the number of applications of the δ' function is bounded by $|t|$. In addition, every application of δ' involves again the operations over multisets that we have referred before.

Consequently, the maximal time needed to run the algorithm is $\mathcal{O}(|t|^2)$. □

4 Recognizing Membrane Structures with Multiset Tree Automata

Recently, in [6], a way to generate trees by membrane systems has been proposed. Initially, any membrane structure can be represented by a tree taking the membrane structure as a hierarchical order between regions. Freund *et al.* [6] have taken advantage of a variant of P systems with active membranes and string objects. Active membranes have an electrical charge (*polarization*) together with a set of rules that allow the membranes to change polarizations, move objects (strings), dissolving the membranes, 2-dividing the membranes, etc. In [6] it is proved that any recursively enumerable tree language can be generated by a P system.

Here, we propose a way to recognize two identical membrane structures by taking advantage of tree representations. For example, let us see Figure 3, in which we represent a membrane structure with different trees.

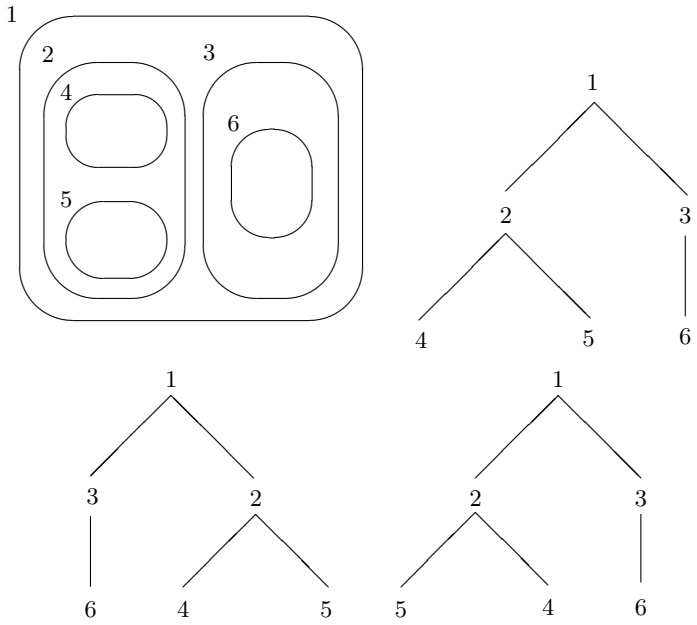

Fig. 3. A membrane structure together with different representations by trees.

Obviously, the initial order of a membrane structure can be fixed. Anyway, whenever the system evolves (membrane dissolving, division, creation, etc.) this order can be at least somehow ambiguous. Furthermore, the initial order of a P system is only a naming convention given that the membrane structure of any P system can be renamed without changing its behavior due to the parallelism (observe that if the P system were sequential, then the ordering could be important for the final output).

The representation by trees could be essential for the analysis of the dynamic behavior of P systems. Whenever we work with trees to represent the membrane structure of a given P system, we can find a *mirroring effect*. Again, take a look at Figure 3: the three different trees proposed for the membrane structure have a mirroring property, that is, some nodes at a given level of the tree have been permuted together (or not) with their descendants.

The method that we propose to establish if two membrane structures μ and μ' are identical is based on the algorithm from Figure 2. First, we represent μ and μ' by t and s, respectively. Then, we apply the proposed algorithm and, if $t \bowtie s$, then we can affirm that μ and μ' are identical.

5 Conclusions and Future Work

We have proposed a new model of tree automata to evaluate if membrane structures of P systems are identical. This is a first step to evaluate the number of membrane rules needed to transform a membrane structure into a different one. The number of rules needed, if so, establishes an editing distance between P systems by taking into account only membrane modifications. Finally, this measure will provide new definitions about *structural confluence* in P systems, that is, structural agreement during evolution.

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, G. Păun, G. Rozenberg, A. Salomaa, eds.: *Multiset Processing. Mathematical, Computer Science and Natural Computing Points of View*, LNCS 2235, Springer-Verlag, Berlin, 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 first, 2002.
4. E. Csuhaj-Varjú, A. Di Nola, G. Păun, M. Pérez-Jiménez, G. Vaszil: Editing configurations of P systems. In the present volume, 131–154.
5. E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana: Multiset automata. In [2], 69–83.

6. R. Freund, M. Oswald, A. Păun: P systems generating trees. In *Pre-proceedings of Fifth Workshop on Membrane Computing (WMC5)*, Milano, 2004 (G. Mauri, G. Păun, C. Zandron, eds.), 221–232.
7. F. Gécseg, M. Steinby: Tree languages. In [14], 1–69.
8. M. Kudlek, C. Martín-Vide, G. Păun: Towards a formal macroset theory. In [2], 123–133.
9. D. López, José M. Sempere, P. García: Error correcting analysis for tree languages. *International Journal of Pattern Recognition and Artificial Intelligence*, 14, 3 (2000), 357–368.
10. A. Păun: On P systems with active membranes. In *Proc. of the First Conference on Unconventionals Models of Computation (UMC2K)*, Brusells, 2000, 187–201.
11. G. Păun: *Membrane Computing. An Introduction.* Springer-Verlag, Berlin, 2002.
12. G. Păun, Y. Suzuki, H. Tanaka, T. Yokomori: On the power of membrane division on P systems. *Theoretical Computer Sci.*, 324, 1 (2004), 61–85
13. R.L. Rivest T.H. Cormen, C.E. Leiserson, C. Stein. *Introduction to Algorithms.* MIT Press and McGraw-Hill, second edition, 2001.
14. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, Springer-Verlag, Berlin, 1997.
15. A. Syropoulos: Mathematics of multisets. In [2], 347–358.