Translating Multiset Tree Automata into P Systems*

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
jsempere@dsic.upv.es

Abstract. In this work we propose a translation scheme to obtain P systems with membrane creation rules from transitions of multiset tree automata (MTA).

1 Introduction

The relation between membrane structures and multiset tree automata (MTA) has been explored in previous works. For instance, in [8] we introduced multiset tree automata, and in [5] this model was used to calculate editing distances between membrane structures. A method to infer multiset tree automata from membrane observations was presented in [9], while in [10] different families of membrane structures were characterized by using multiset tree automata.

In this work we propose a translation scheme to obtain membrane rules from MTA transitions. The advantages of this approach are clear, so we can implement a computer tool to automatically obtain membrane creation rules from a set of trees that model the desired behavior of the P system structure according to [9].

2 Notation and Definitions

In the sequel we provide some concepts from formal language theory, membrane systems and multiset processing. We suggest the books [7], [6] and [1] to the reader.

Multisets

First, we provide some definitions from multiset theory as exposed in [11].

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

* Work supported by the Spanish Ministerio de Ciencia e Innovación under project TIN2007-60769.

D. Corne et al. (Eds.): WMC9 2008, LNCS 5391, pp. 394–402, 2009.

We say that A is *empty* if for all $a \in D$, $f(a) = 0$, and $A = B$ if the multiset $(A \ominus B) \oplus (B \ominus A)$ is empty.

The size of a multiset M is the number of elements that it contains and it is denoted by $|M|$ (observe that we take into account the multiplicities of every element). 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 $\sum_{a\in D} f(a) = n$ (observe that, in this case, $\langle D, f\rangle$ should be finite).

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 $\Psi : D^* \to \mathbb{N}^n$ 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, $1 \leq j \leq n$. Finally, in the following, we work with strings representing multisets. So, the multiset represented by x is the multiset with elements that appear in x and multiplicities according to $\Psi(x)$.

P Systems

We introduce basic concepts from membrane systems taken from [6]. A transition 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*),
- 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 a symbol not in V that defines the *membrane dissolving action*. From now on, we denote the set *tar* by $\{here, out, in_k \mid 1 \leq k \leq m\}$,
- 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 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 is denoted

by $N(\Pi)$. Obviously, both sets $L(\Pi)$ and $N(\Pi)$ are defined only for *halting computations*.

Many kinds of rules have been proposed in P systems for creation, division and modification of membrane structures. There have been several works in which these rules have been investigated or they have been employed for different purposes.

In the following, we enumerate two kinds of rules which are able to modify the membrane structure.

1. Division: $[_h a]_h \rightarrow [_h [_{h_1} a_1]_{h_1} [_{h_2} a_2]_{h_2} \cdots [_{h_p} a_p]_{h_p}]_h$. The object a in region h is transformed into objects $a_1, a_2, \cdots, a_p$. Then, p new regions are created inside h with labels $h_1, h_2, \cdots, h_p$, and the new objects are communicated to the new regions. This rule is a generalization of the 2-division rule proposed in different works.
2. Creation: $a \rightarrow [_h b]_h$. A new region is created with label h and the object a is transformed into object b which is communicated to the new region.

The power of P systems with the previous operations and other ones (e.g., *exocytosis, endocytosis*, etc.) has been widely studied in the membrane computing area. Given that the previous operations can modify the membrane structure of a P system Π during the computation, we denote by $str(\Pi)$ the set of membrane structures (trees) that eventually are reached by Π during its computation. Observe that this definition was used in [3], in order to define tree languages generated by Π systems. In such case, only the membrane structures obtained after halting were considered.

In this work we introduce a new kind of rules which allow the creation of different regions in only one step: n-creation. Formally, a n-creation rule is defined as $a \rightarrow w_0 [_{h_1} w_1]_{h_1} [_{h_2} w_2]_{h_2} \cdots [_{h_n} w_n]_{h_n}$. The meaning of this rule is that the object a is transformed into objects $w_0, w_1, \cdots, w_n$. Then, n new regions are created with (probably repeated) labels $h_1, h_2, \cdots, h_n$, and the new objects are placed in the new regions. This rule is a generalization of the creation rule proposed in different works; we refer to [6] for references. Observe that, under our point of view, the use of a n-creation rule is equivalent to the use of a creation rule together with a division rule and then a dissolving rule in order to erase the external region.

Tree Automata and Tree Languages

Now, we introduce some concepts from tree languages and automata as exposed in [2,4]. 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 $\{\sigma \in V \mid (\sigma, n) \in r\}$. We denote by $maxarity(V)$ the maximum integer k such that $V_k \neq \emptyset$.

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 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 formed by 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 $dom(t)$.

Definition 1. *A* deterministic finite 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 refer to deterministic finite tree automata simply as *tree automata*. We suggest [2,4] 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 automaton A cannot accept any tree of depth zero.

Multiset Tree Automata and Mirrored Trees

We extend some definitions of tree automata and tree languages over multisets. We introduce the concept of multiset tree automaton and then we 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 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 is 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 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. *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:*

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

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

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

Definition 3. *Let* $A = (Q, V, \delta, F)$ *be a tree automaton. The multiset tree automaton induced by* A *is 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

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

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

The following two results formally relate tree automata and multiset tree automata.

Theorem 1. *(Sempere and López, [8]) 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, [8]) 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 [8]. Informally speaking, two trees are 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 4. *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* $\langle s^1, s^2, \ldots, s^k \rangle \in perm(\langle s_1, s_2, ..., s_n \rangle)$ *such that* $\forall 1 \leq i \leq n\ t_i \bowtie s^i$

Theorem 2. *(Sempere and López, [8]) 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, [8]) 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 $t \bowtie s$, $s \in L(MA)$.*

3 From MTA Transitions to Membrane Rules

In this section, we propose a translation scheme to obtain P systems from MTA. The relation between the input and the output of the scheme is showed at the end of this section. In addition, we show that the obtained P system generates membrane structures which can be represented by trees that the input MTA accepts. First, we provide a couple of examples that give some intuition in the scheme that we propose later.

Example 1. Consider the multiset tree automaton with transitions:

$$\delta(\sigma, aa) = q_1,\ \delta(\sigma, a) = q_2,\ \delta(\sigma, aq_2) = q_2,\ \delta(\sigma, q_1q_1) = q_1,\ \delta(\sigma, aq_2q_1) = q_3 \in F.$$

Then, the following P system is able to produce, during different computations, a set of membrane structures such that the set of trees induced by them are the set of trees accepted by the MTA.

$\Pi = (\{a,b\}, \{a,b\}, \emptyset, [\,]_{q_3}, b, \emptyset, \emptyset, (R_{q_3}, \emptyset), (R_{q_2}, \emptyset), (R_{q_1}, \emptyset), \infty)$, where
$R_{q_3} = \{b \to a[_{q_2} b]_{q_2} [_{q_1} b]_{q_1}\}$,
$R_{q_2} = \{b \to a[_{q_2} b]_{q_2}; b \to a\}$, and
$R_{q_1} = \{b \to aa; b \to [_{q_1} b]_{q_1} [_{q_1} b]_{q_1}\}$

Observe that we have made a top-down design, in which we start by analyzing the final states (in this case q_3) and then we obtain the ancestors of every state according with δ by using membrane creation and membrane division.

In the following example the number of final states is greater than one.

Example 2. Let us take the MTA defined by the following transitions

$$t_1 : \delta(\sigma, aa) = q_1,\ t_2 : \delta(\sigma, bb) = q_2,\ t_3 : \delta(\sigma, q_2 q_2) = q_2,\ t_4 : \delta(\sigma, q_1 q_1) = q_1,\ t_5 : \delta(\sigma, q_2 q_1) = q_3.$$

The following P system is associated to the previous MTA. Observe that we have added a superscript to every membrane rule according to every MTA transition.

$\Pi = (\{a,b,c\}, \{a,b,c\}, \emptyset, [\,]_0, c, \emptyset, \emptyset, \emptyset, (R_0, \emptyset), (R_{q_2}, \emptyset), (R_{q_1}, \emptyset), \infty)$, where
$R_0 = \{c \to aa^{(1)}; c \to bb^{(2)}; c \to [_{q_2} c]_{q_2} [_{q_2} c]^{(3)}_{q_2}; c \to [_{q_1} c]_{q_1} [_{q_1} c]^{(4)}_{q_1};$
$c \to [_{q_2} c]_{q_2} [_{q_1} c]^{(5)}_{q_1}\}$
$R_{q_2} = \{c \to bb^{(2)}; c \to [_{q_2} c]_{q_2} [_{q_2} c]^{(3)}_{q_2}\}$
$R_{q_1} = \{c \to aa^{(1)}; c \to [_{q_1} c]_{q_1} [_{q_1} c]^{(4)}_{q_1}\}$

We propose Algorithm 1 as a translation scheme from MTA to P systems. The main step of the proposed algorithm, is step 5 which uses a transformation $\wp_c$ over $M_\Psi(\delta)$. We formally define the transformation $\wp_c$ as follows: $\wp_c(p_1 \cdots p_k) = \wp_c(p_1) \cdots \wp_c(p_k)$, where

$$\wp_c(p_i) = \begin{cases} p_i & \text{if } p_i \in \Sigma_0 \\ [_{p_i} c]_{p_i} & \text{if } p_i \in Q \end{cases}$$

Now, we can formally prove the correctness of the proposed algorithm through the following result.

Proposition 1. *Algorithm 1 obtains a P system Π from the input MTA A such that $str(\Pi) = L(A)$.*

Proof. The key step in the proposed algorithm is step 5. Observe that the step 5, ensures that if $\delta(\sigma, p_1 \cdots p_k) = q_j$ then the region R_j holds a rule such that for every state p_l in the ancestors of q_j, according to the transition, a new region $[_{q_l} c]_{q_l}$ is created. On the other hand, every symbol $a \in p_1 \cdots p_k$ is created in region R_j. So, if the structure $\sigma(p_1 \cdots p_k)$ (or any mirrored one) is reduced to

Algorithm 1. A translation scheme from MTA to P systems

Input: A MTA $A = (Q, \Sigma, \delta, F)$
Output: A P system $\Pi = (V, T, \emptyset, []_0, c, \emptyset, \cdots, \emptyset, (R_0, \rho_0), \cdots, (R_m, \rho_m), i_0)$ such that $str(\Pi) = L(A)$
Method:

1. $V = T = \Sigma_0 \cup \{c\}$ such that $c \notin \Sigma_0$
2. $m = |Q|$
3. $\rho_i = \emptyset$ $0 \leq i \leq |Q|$
4. $R_i = \emptyset$ $0 \leq i \leq |Q|$
5. For every transition in δ such that $\delta(\sigma, p_1 \cdots p_k) = q_j$
   If $q_j \in F$
   then Add to R_0 the rule $c \rightarrow \wp_c(p_1 \cdots p_k)$
   Add to R_j the rule $c \rightarrow \wp_c(p_1 \cdots p_k)$
6. **Return(Π)**

EndMethod.

the state q_j in the MTA A, the structure $\wp_c(p_1 \cdots p_k)$ is created in the P system Π inside the region R_j. In addition, if $q_j \in F$ then all the (mirrored) trees reduced to q_j are accepted by A, so this is the reason why all these structures are inside the skin region R_0.

On the other hand, observe that the unique object which can create new membranes is c which does not belong to Σ_0. We have introduced c because the rest of symbols are just leaves in the trees accepted by A. So, once any of the leaves appears, it remains in the region as a an object that cannot evolve anymore. Finally, the objects c disappear when all the leaves of the trees are created.

Another aspect that we take under our consideration is the efficiency of the proposed algorithm. We analyze its complexity time through the following result.

Proposition 2. *Algorithm 1 runs in polynomial time with respect to the size of the input MTA A.*

Proof. Again, the main step of the proposed algorithm is step 5. Here, we make as many operations as the number of δ transitions. For every transition, we must evaluate the transformation $\wp_c$ which is quadratic with the size of the ancestors of every state and the union of $|Q|$ and Σ_0. This holds a quadratic running time for Algorithm 1.

4 Conclusions and Future Work

In this work we have proposed a full translation scheme from MTA to P systems. The proposed algorithm correctly and efficiently performs the translation task. This scheme gives a formal proof for the relation between the structures generated by the P system with membrane n-creation rules (or membrane creation plus membrane division) and the trees accepted by MTA. This result was pointed out in previous works such as [5,8,9,10].

Actually, we are developing a computer tool that holds the proposed translation scheme. This tool will help to analyze the membrane dynamics in P systems by using the results proposed in [9]. Furthermore, we will be able to propose initial P systems based only in the membrane structures we want to generate which will be enriched later with the corresponding evolution and communication rules.

On the other hand, a topic which has been investigated in previous works is the relationship between MTA and P systems. We can study in depth some aspects of the P systems by only observing the membrane dynamics. This study can be achieved by characterizing different MTA classes as was proposed in [10]. We think that we must keep on this research in order to get a complex picture of different P systems and their relations by using only MTA.

Acknowledgements. The author is grateful to the reviewers for sharp remarks and suggestions made to this work. The author is most indebted to Mario J. Pérez-Jiménez and Gheorghe Păun for comments on n-creation rules during the *9th Workshop on Membrane Computing* at Edinburgh.

References

1. Calude, C.S., Păun, G., Rozenberg, G., Salomaa, A. (eds.): Multiset Processing. LNCS, vol. 2235. Springer, Heidelberg (2001)
2. Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications (1997) (October 1st, 2002), `http://www.grappa.univ-lille3.fr/tata`
3. Freund, R., Oswald, M., Păun, A.: P systems generating trees. In: Mauri, G., Păun, G., Jesús Pérez-Jímenez, M., Rozenberg, G., Salomaa, A. (eds.) WMC 2004. LNCS, vol. 3365, pp. 309–319. Springer, Heidelberg (2005)
4. Gécseg, F., Steinby, M.: Tree languages. In: Handbook of Formal Languages, vol. 3, pp. 1–69. Springer, Heidelberg (1997)
5. López, D., Sempere, J.M.: Editing distances between membrane structures. In: Freund, R., et al. (eds.) WMC 2005. LNCS, vol. 3850, pp. 326–341. Springer, Heidelberg (2006)
6. Păun, G.: Membrane Computing. An Introduction. Springer, Heidelberg (2002)
7. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer, Heidelberg (1997)
8. Sempere, J.M., López, D.: Recognizing membrane structures with tree automata. In: Gutiérrez Naranjo, M.A., et al. (eds.) Proc. 3rd Brainstorming Week on Membrane Computing, Fénix Editora, Sevilla, pp. 305–316 (2005)
9. Sempere, J.M., López, D.: Identifying P rules from membrane structures with an error-correcting approach. In: Hoogeboom, H.J., et al. (eds.) WMC 2006. LNCS, vol. 4361, pp. 507–520. Springer, Heidelberg (2006)
10. Sempere, J.M., López, D.: Characterizing membrane structures through multiset tree automata. In: Eleftherakis, G., et al. (eds.) WMC 2007. LNCS, vol. 4860, pp. 428–437. Springer, Heidelberg (2007)
11. Syropoulos, A.: Mathematics of multisets. In: [1], pp. 347–358