Computer Science Journal of Moldova, vol.**18**, no.2(53), 2010

Membrane Systems Languages Are Polynomial-Time Parsable

Artiom Alhazov Constantin Ciubotaru Sergiu Ivanov
Yurii Rogozhin

Abstract

The focus of this paper is the family of languages generated by transitional non-cooperative P systems without further ingredients. This family can also be defined by so-called time yields of derivation trees of context-free grammars. In this paper we prove that such languages can be parsed in polynomial time, where the degree of polynomial may depend on the number of rules and on the size of the alphabet.

1 Introduction

Membrane computing is a theoretical framework of parallel distributed multiset processing. It has been introduced by Gheorghe Păun in 1998, and has been an active research area; see [6] for the comprehensive bibliography and [4],[3] for a systematic survey. Membrane systems are also called P systems.

The configurations of membrane systems (with symbol objects) consist of multisets over a finite alphabet, distributed across a tree structure. Therefore, even such a relatively simple structure as a word (i.e., a sequence of symbols) is not explicitly present in the system. To speak of languages as sets of words, one first needs to represent them in membrane systems, and there are a few ways to do it.

One of the most elegant ways is to do all the processing by multisets, and regard the order of sending the objects in the environment as their order in the output word. In case of ejecting multiple symbols in the same step, the output word is formed from any of their permutations. One can say that this approach also needs an implicit observer, but at least this observer only inspects the environment and it is, in some sense, the simplest possible one.

Following [2], in this paper we are interested in the case when the rules are non-cooperative, i.e., all objects evolve independently. A number of results have been established in [2]. For instance, it was shown that one membrane is enough, and a characterization of this family was given via derivation trees of context-free grammars. Next, three normal forms were given for the corresponding grammars. It was than shown that the membrane systems language family lies between regular and context-sensitive families of languages, and it is incomparable with linear and with context-free languages. Then, the lower bound was strengthened to $REG \bullet \texttt{Perm}(REG)$. An example of a considerably more "difficult" language was given than the lower bound mentioned above. The membrane systems language family was also shown to be closed under union, permutations, erasing/renaming morphisms. It is not closed under intersection, intersection with regular languages, complement, concatenation or taking the mirror image.

In attempt to lower the known upper bound (semilinear context-sensitive) of these languages, we show here that the word membership problem can be solved in polynomial time.

2 Definitions

Consider a finite set V. The set of all words over V is denoted by V^*, the concatenation operation is denoted by $\bullet$ (which is written only when necessary) and the empty word is denoted by λ. Any set $L \subseteq V^*$ is called a language. For a word $w \in V^*$ and a symbol $a \in V$, the number of occurrences of a in w is written as $|w|_a$. We write $w[i]$ to denote the i-th symbol of w, $1 \leq i \leq |w|$. The permutations of a word $w \in V^*$ are $\texttt{Perm}(w) = \{x \in V^* \mid |x|_a = |w|_a \forall a \in V\}$. We denote the set of all permutations of the words in L by $\texttt{Perm}(L)$, and we extend this notation to families of languages. We use FIN, REG, LIN, CF, MAT, CS, RE to denote finite, regular, linear, context-free, matrix without appearance checking and with erasing rules, context-sensitive and recursively enumerable families of languages, respectively. The family of languages generated by extended (tabled) interactionless L systems is denoted by $E(T)0L$. Notation $SLIN$ stands for the semilinear languages. We denote by **P** the family of languages recognizable by Turing machines in polynomial time. For more formal language preliminaries, we refer the reader to [5].

A multiset over V is a mapping $M : V \to \mathbb{N}$; $M(a)$ is multiplicity of a in M. For $V = \{a_1, \cdots, a_m\}$, we may write M as $\left\{a_1^{M(a_1)}, \cdots, a_m^{M(a_m)}\right\}$, omitting missing elements. The size $|M|$ of a multiset is $\sum_{i=1}^{m} M(a_i)$. We use the extension of the set notations to multisets; for instance, $M_1 \subseteq M_2$, $M_1 \cup M_2$ and $M_1 \setminus M_2$ mean $M_1(a) \leq M_2(a)$, $M_1(a) + M_2(a)$ and $\max(M_1(a) - M_2(a), 0)$ for the multiplicities of all symbols a, respectively. Multisets in membrane computing are typically represented by strings; in this paper we use the set notations described above, to be able to distinguish between multisets and strings.

2.1 Transitional P systems

A membrane system is defined by a construct

$$\begin{array}{rcl}
\Pi & = & (O, \mu, w_1, \cdots, w_m, R_1, \cdots, R_m, i_0), \text{ where} \\
O & & \text{is a finite set of objects,} \\
\mu & & \text{is a hierarchical structure of membranes,} \\
w_i & & \text{is the initial multiset in region } i,\ 1 \leq i \leq m, \\
R_i & & \text{is the set of rules of region } i,\ 1 \leq i \leq m, \\
i_0 & & \text{is the output region.}
\end{array}$$

The membranes are bijectively labeled by $1, \cdots, m$, the interior of each membrane defines a region; the environment is referred to as region 0. When languages are considered, $i_0 = 0$ is assumed.

The rules of a membrane system have the form $u \to v$, where u is a non-empty multiset over O and v is a multiset over $(O \times Tar)$. The target indications from $Tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq m\}$ are written as a subscript, and target *here* is typically omitted. In case of non-cooperative rules, u is a multiset of size 1.

The rules are applied in maximally parallel way: no further rule should be applicable to the idle objects. In case of non-cooperative systems, the concept of maximal parallelism is the same as in L systems: all objects evolve by the associated rules in the corresponding regions (except objects a in regions i such that R_i does not contain any rule $a \to u$, but these objects do not contribute to the result). The choice of rules is non-deterministic.

A configuration of a P system is a construct which contains the information about the hierarchical structure of membranes as well as the contents of every membrane at a definite moment of time. The process of applying all rules which are applicable in the current configuration and thus obtaining a new configuration is called a transition. A sequence of transitions is called a computation. The computation halts when such a configuration is reached that no rules are applicable. The result of a (halting) computation is the *sequence* of objects sent to the environment (all the permutations of the symbols sent out in the same time are considered). The language $L(\Pi)$ generated by a P system Π is the union of the results of all computations. The family of languages generated by non-cooperative transitional P systems with at most m membranes is denoted by $LOP_m(ncoo, tar)$. If the number of membranes is not bounded, m is replaced by $*$ or omitted. If the target indications of the form in_j are not used, tar is replaced by out.

Example 1 *To illustrate the concept of generating languages, consider the following P system:*

$$\Pi = (\{a, b, c\}, [_1 \ \ ]_1, \{a^2\}, \left\{\{a\} \to \emptyset, \{a\} \to \{a, b_{out}, c_{out}^2\}\right\}, 0).$$

Each of the two symbols a has a non-deterministic choice whether to be erased or to reproduce itself while sending a copy of b and two copies of c into the environment. Therefore, the contents of region 1 can remain a^2 for an arbitrary number $m \geq 0$ of steps, and after that at least one copy of a is erased. The other copy of a can reproduce itself for another $n \geq 0$ steps before being erased. Each of the first m steps, two

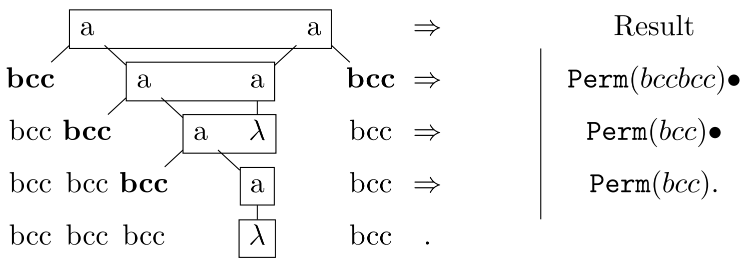


Figure 1. An example of a computation of a P system from Example 1. The lines are only used to hint how the rules are applied.

copies of b and four copies of c are sent out, while in each of the next n steps, only one copy of b and two copies of c are ejected. Therefore, $L(\Pi) = (\texttt{Perm}(bccbcc))^*(\texttt{Perm}(bcc))^*$.

3 Parsability

We first recall a few existing results.

Lemma 1 *Random access machines can be simulated by Turing machines with polynomial slowdown.*

This result will lead to a much simpler proof of the main result.

Lemma 2 *[2] $LOP_*(ncoo, tar) = LOP_1(ncoo, out)$.*

This result means one membrane is enough. Such membrane systems only have one working region, and the destination of the objects in right hand side of the rules may only be *here* and *out*.

Lemma 3 *[2] Any non-cooperative P system can be transformed into an equivalent one such that all objects evolve by some rules (objects not participating in left-hand side of any rule are never produced).*

This condition implies that the system only halts if there are no objects inside the system. Hence, the evolution of any object inside the system eventually leads to some number (possibly zero) of objects in the environment.

Lemma 4 *([2]) Any non-cooperative P system can be transformed into an equivalent one such that the initial contents is $w_1 = \{S\}$, and*

- *S does not appear in the right-hand side of any rule, and*
- *R_1 has no erasing rules, except possibly $\{S\} \to \emptyset$.*

This result means that no object can be erased, except the axiom which may only be erased immediately.

We now proceed to the main result.

Theorem 1 *$LOP_*(ncoo, tar) \subseteq$* **P**.

Proof. The proof consists of three parts. First, a few known results are used to simplify the statement of the theorem. Second, a finite-state automaton (with transitions labeled by multisets of terminals) of polynomial size is constructed. Third, acceptance problem is reduced to a search problem in a graph of a polynomial size.

Thanks to Lemma 1, the rest of the proof can be explained at the level of random access machines.

Due to Lemma 2, we assume that an arbitrary membrane system language L is given by a one-membrane system $\Pi = ([_1\ ]_1, O, w_1, R_1)$.

It is known from Lemma 3 that the condition specified in it does not restrict the generality. Hence, from now we assume that every object A inside the system corresponds to at least one rule that rewrites A.

Without restricting generality, we also assume the normal form specified in Lemma 4. In this case, it is clear that if $w \in T^n$, then during any computation of Π generating w, the number of objects inside the system can never exceed $\max(n, 1)$.

We now build a finite automaton $A = (Q, \Sigma, q_0, \delta, F)$ such that any word $w' \in T^{\leq n}$ is accepted by A if and only if $w' \in L(\Pi)$. Accepting by an automaton with transitions labeled by multisets is understood as follows: a transition labeled by a multiset of weight k can be followed if the multiset composed of the next k input symbols equals the transition label; in this case these input symbols are read.

We define Q as the set of multisets of at most $\max(n, 1)$ objects, Σ as the set of multisets of at most n objects, q_0 is the singleton multiset $\{S\}$, and $F = \{\emptyset\}$. It only remains to define the transition mapping δ of A. We say that $q' \in \delta(q, s)$ if $[_1\ q\ ]_1 \Rightarrow [_1\ q'\ ]_1 s$. It is known (see, e.g., [1]) that computing all transitions from a configuration with k objects takes polynomial time with respect to k; here, $k \leq \max(n, 1)$ (and the degree of such a polynomial does not exceed $|R_1|$), and, moreover, the number of configurations reachable in one step is also polynomial.

Notice that $|Q|$ is polynomial with respect to n (and the degree of such a polynomial does not exceed $|O| + 1$).[1] Hence, building A from n and Π can be done in polynomial time, and, moreover, the size of the description of A is also polynomial. Of course, it is sufficient to only examine the reachable states of A.

Running each transition $q' \in \delta(q, s)$ of A on w can be actually done in time $O(|s|)$; however, there are two problems. Firstly, A is non-deterministic, and secondly, A may have transitions labeled by an empty multiset, and removing empty multiset transitions or non-determinism might need too much time or space, or even increase its size too much. Instead, we reduce parsing by A to a graph reachability problem.

Consider a graph $\Gamma = (V, U)$, where $V = \{0, \cdots, n\} \times Q$ and U consists of such transitions $((i, q), (j, q'))$ that $i \leq j$ and $q' \in \delta(q, s)$, where s equals the multiset consisting of $w[i + 1], \cdots, w[j]$.

Finally, $w \in L = L_t(G)$ if and only if $w \in L(A)$, and $w \in L(A)$ if and only if there is a path from $(0, q_0)$ to (n, e) in Γ. Note: alternatively, search in A incrementally by prefixes of w. $\square$

[1] Indeed, multisets of size $\leq n$ over O bijectively correspond to multisets of size exactly n over $O \cup \{\lambda\}$. Let $|O| = m$. Moreover, multisets of size n over $O \cup \{\lambda\}$ correspond to n-combinations of $m+1$ possible elements with repetition. For $n > 0$, their number is $|Q| = \binom{n+m}{n} \leq n^{m+1}$.

It is known from [2] that membrane systems language family is included in the family of context-sensitive languages, see also Lemma 4. In [2] one also claims that membrane systems language family is **semilinear**. No formal proof is given, but there is an almost immediate observation that such language is letter-equivalent to that generated by the context-free language with the same rules (languages are letter-equivalent if for every word in one of them there is a word in the other one with the same multiplicities of all symbols; indeed, the difference is only in the order of output). Semilinearity thus follows from Parikh theorem. By Theorem 1, we improve the upper bound:

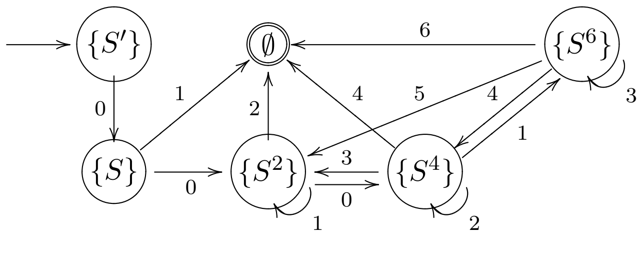
Corollary 1 $LOP_*(ncoo, tar) \subseteq CS \cap SLIN \cap \mathbf{P}$.

4 An Example

Consider a word $w = babbaa$ and a P system

$$\begin{aligned} \Pi &= ([_1\ ]_1, \{S', S, a, b\}, \{S'\}, R), \text{ where} \\ R &= \{p : \{S'\} \to \{S\},\ q : \{S\} \to \{S^2\},\ r : \{S\} \to \{a, b\}_{out}\}. \end{aligned}$$

Only objects S, S' are productive inside the system, and only objects a, b may be sent outside. Since $|w| = 6$, we only need to examine multisets over S, S' of size up to 6 elements (28 in total). However, out of them only $\{S'\}$, $\{S\}$, $\emptyset$, $\{S^2\}$, $\{S^4\}$, $\{S^6\}$ are reachable. The finite automaton would look as follows (for simplicity of the picture, we wrote i instead of $\{a^i, b^i\}$ as labels):



We now check the word w:

- states after reading λ: $\{S'\}$, $\{S\}$, $\{S^2\}$, $\{S^4\}$;

- states after reading *ba*: $\emptyset$, $\{S^2\}$, $\{S^4\}$, $\{S^6\}$;
- states after reading *babbaa*: $\emptyset$, $\{S^4\}$. The input is accepted.

5 Conclusions

We have shown that there exists an algorithm deciding the word membership problem of membrane systems languages in polynomial time with respect to the length of the word. The degree of such polynomial may depend on the number of rules and on the size of the alphabet. Hence, the position of the membrane systems language family in the language family hierarchy is between $REG \bullet \texttt{Perm}(REG)$ and $CS \cap SLIN \cap \mathbf{P}$.

Acknowledgments Artiom Alhazov gratefully acknowledges the support of the Japan Society for the Promotion of Science and the Grant-in-Aid for Scientific Research, project 20·08364. All authors acknowledge the support by the Science and Technology Center in Ukraine, project 4032.

References

[1] A. Alhazov: Maximally Parallel Multiset-Rewriting Systems: Browsing the Configurations. In: M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan: RGNC report 01/2005, University of Seville, *Third Brainstorming Week on Membrane Computing*, Fénix Editora, Sevilla, 2005, 1–10.

[2] A. Alhazov, C. Ciubotaru, Yu. Rogozhin, S. Ivanov: The Family of Languages Generated by Non-Cooperative Membrane Systems. In: M. Gheorghe, Th. Hinze, Gh. Păun: *Preproceedings of the Eleventh Conference on Membrane Computing, CMC11*, Jena, Verlag ProBusiness Berlin, 2010, 37–51, and *Lecture Notes in Computer Science* **6501**, to appear.

[3] Gh. Păun, G. Rozenberg, A. Salomaa, Eds.: *Handbook of Membrane Computing.* Oxford University Press, 2010.

[4] Gh. Păun: Membrane Computing. An Introduction, Springer, 2002.

[5] G. Rozenberg, A. Salomaa, Eds.: *Handbook of Formal Languages*, vol. 1-3, Springer, 1997.

[6] P systems webpage. `http://ppage.psystems.eu/`

A. Alhazov[1,2], C. Ciubotaru[1], S. Ivanov[1,3],
Yu. Rogozhin[1]

Received November 1, 2010

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova
E–mail: {`artiom, chebotar, sivanov, rogozhin`}`@math.md`

[2] FCS, Department of Information Engineering
Graduate School of Engineering
Hiroshima University,
Higashi-Hiroshima 739-8527 Japan

[3] Faculty of Computers,
Informatics and Microelectronics,
Technical University of Moldova,
Ştefan cel Mare 168, Chişinău MD-2004 Moldova