Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 3, pp. 206-213

Dictionary Search and Update by P Systems with String-Objects and Active Membranes

Artiom Alhazov, Svetlana Cojocaru, Ludmila Malahova, Yurii Rogozhin

Artiom Alhazov, Svetlana Cojocaru, Ludmila Malahova, Yurii Rogozhin
Institute of Mathematics and Computer Science, Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
E-mail: {artiom,sveta, mal, rogozhin}@math.md

Artiom Alhazov
IEC, Department of Information Engineering, Graduate School of Engineering
Hiroshima University, Higashi-Hiroshima 739-8527 Japan

Yurii Rogozhin
Rovira i Virgili University, Research Group on Mathematical Linguistics
Av. Catalunya, 35, Tarragona 43002 Spain
E-mail: yrogozhin@yahoo.com

Received: April 5, 2009
Accepted: May 30, 2009

Abstract: Membrane computing is a formal framework of distributed parallel computing. In this paper we implement the work with the prefix tree by P systems with strings and active membranes. We present the algorithms of searching in a dictionary and updating it implemented as membrane systems. The systems are constructed as reusable modules, so they are suitable for using as sub-algorithms for solving more complicated problems.

Keywords: Membrane computing, P systems, active membranes, dictionary, prefix tree

1 Introduction

Solving most problems of natural language processing is based on using certain linguistic resources, represented by corpora, lexicons, etc. Usually, these collections of data constitute an enormous volume of information, so processing them requires much computational resources. A reasonable approach for obtaining efficient solutions is that based on applying parallelism; this idea has been promoted already in 1970s. For instance, the possibilities of applying massive parallelism in Machine Translation are considered in [5, 2]. Many of the stages of text processing (from tokenization, segmentation, lematizing to those dealing with natural language understanding) can be carried out by parallel methods. This justifies the interest to the methods offered by the biologically inspired models, and by membrane computing in particular.

However, there are some issues that by their nature do not allow complete parallelization, yet exactly they are often those “computational primitives" that are inevitably used during solving major problems, like the elementary arithmetic operations are always present in solving difficult computational problems. Among such “primitives" in the computational linguistics we mention handling of the dictionaries, e.g., dictionary lookup and dictionary update. Exactly these problems constitute the subject of the present paper. In our approach we speak about dictionary represented by a prefix tree.

P (membrane) systems are a convenient framework of describing computations on trees. Since membrane systems are an abstraction of living cells, the membranes are arranged hierarchically, yielding a tree structure.

2 Definitions

Membrane computing is a recent domain of natural computing initiated by Gh. Păun in [12]. The components of a membrane system are a cell-like membrane structure, in the regions of which one places multisets of objects which evolve in a synchronous maximally parallel manner according to given evolution rules associated with the membranes. The necessary definitions are given in the following subsection; see also [4] for an overview of the domain and [6] for a comprehensive bibliography.

2.1 Computing by P systems

Let O be a finite set of elements called symbols; then set of words over O is denoted by O^*, and the empty word is denoted by λ.

Definition 1. A P system with string-objects and input is a tuple

$\Pi = \big(O, \Sigma, H, E, \mu, M_1, \cdots, M_p, R, i_0\big)$, where:

- O is the working alphabet of the system (the objects are strings over O),
- Σ is an input alphabet,
- H is an alphabet whose elements are called labels, i_0 identifies the input region,
- E is the set of polarizations,
- μ is a membrane structure (a rooted tree) consisting of p membranes injectively labeled by elements of H,
- M_i is an *initial* multiset of strings over O associated with membrane i, $1 \leq i \leq p$,
- R is a finite set of rules defining the behavior of objects from O^* and of membranes labeled by elements of H.

A configuration of a P system is its "snapshot", i.e., the current membrane structure and the multisets of string-objects present in regions of the system. The initial configuration is $C_0 = (\mu, M_1, \cdots, M_p)$. Each subsequent configuration C' is obtained from the previous configuration C by maximally parallel application of rules to objects and membranes. This is denoted by $C \Rightarrow C'$ (no further rules are applicable together with the rules that transform C into C'). A computation is thus a sequence of configurations starting from C_0, respecting relation $\Rightarrow$ and ending in a halting configuration (i.e., such one that no rules are applicable). If M is a multiset of strings over the input alphabet $\Sigma \subseteq O$, then the *initial configuration* of a P system Π with an input M over alphabet Σ and input region i_0 is $(\mu, M_1, \ldots, M_{i_0-1}, M_{i_0} \cup M, M_{i_0+1}, \ldots, M_p)$.

2.2 P systems with active membranes

To speak about P systems with active membranes, we need to specify the rules, i.e., the elements of the set R in the description of a P system. Due to the nature of the problem of this paper, the standard model was generalized in the following:

- Cooperative rules: a rule operates on a substring of an object (otherwise, the system cannot even distinguish different permutations of a string); this feature is represented by a superscript * in the rule types;
- String replication (to return the result without removing it from the dictionary);

- Membrane creation (to add words to the dictionary).

Hence, the rules can be of the following forms:

(a^*) $[\, a \to b \,]_h^e$ for $h \in H, e \in E, a, b \in O^*$- evolution rules
(associated with membranes and depending on the label and the polarization of the membranes, but not directly involving the membranes: the membranes are neither taking part in the application of these rules nor are they modified by them);

(a_r^*) $[\, a \to b || c \,]_h^e$ for $h \in H, e \in E, a, b, c \in O^*$ (as above, but with string replication);

(b^*) $a[\ \ ]_h^{e_1} \to [\, b \,]_h^{e_2}$ for $h \in H, e_1, e_2 \in E, a, b \in O^*$ - communication rules
(an object is introduced into the membrane, possibly modified; the polarization of the membrane can be modified, but not its label);

(c^*) $[\, a \ \,]_h^{e_1} \to [\ \ ]_h^{e_2} b$ for $h \in H, e_1, e_2 \in E, a, b \in O^*$ - communication rules
(an object is sent out of the membrane, possibly modified; also the polarization of the membrane can be modified, but not its label);

(d^*) $[\, a \ \,]_h^e \to b$ for $h \in H, e \in E, a, b \in O^*$ - dissolving rules
(in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

(g^*) $[\, a \ \to [\, b \ \,]_g^{e_2} \,]_h^{e_1}$ for $g, h \in H,\ e_1, e_2 \in E,\ a, b \in O^*$ - membrane creation rules
(an object is moved into a newly created membrane, possibly modified).

Additionally, we will write $\emptyset$ in place of some strings on the right-hand side of the rules, meaning that the entire string is deleted.

The rules of types $(a^*), (a_r^*)$ and (g^*) are considered to only involve objects, while all other rules are assumed to involve objects and membranes mentioned in their left-hand side. An application of a rule consists in replacing a substring described in the left-hand side of a string in the corresponding region (i.e., associated to a membrane with label h and polarization e for rules of types $(a^*), (a_r^*)$ and (d^*), or associated to a membrane with label h and polarization e_1 for rules of type (c^*), or immediately outer of such a membrane for rules of type (b^*)), by a string described in the right-hand side of the rule, moving the string to the corresponding region (that can be the same as the source region immediately inner or immediately outer, depending on the rule type), and updating the membrane structure accordingly if needed (changing membrane polarization, creating or dissolving a membrane). Only the rules involving different objects and membranes can only be applied in parallel; such parallelism is maximal if no further rules are applicable in parallel.

3 Dictionary

Dictionary search represents computing a string-valued function $\{u_i \longrightarrow v_i \mid 1 \le i \le d\}$ defined on a finite set of strings.

We represent such a dictionary by the skin membrane containing the membrane structure corresponding to the prefix tree of $\{u_i \mid 1 \le i \le d\}$, with strings $\$v_i\$'$ in regions corresponding to the nodes associated to u_i. Let A_1, A_2 be the source and target alphabets: $u_i \in A_1^*$, $v_i \in A_2^*$, $1 \le i \le d$. Due to technical reasons, we assume that for every $l \in A_1$, the skin contains a membrane with label l. We also suppose that the source words are non-empty.

For instance, the dictionary $\{\text{bat} \longrightarrow \text{flying}, \text{bit} \longrightarrow \text{stored}\}$ is represented by

$$[\,[\ \,]_a^o[\,[\,[\,\$flying\$'\,]_t^o\,]_a^o[\,[\,\$stored\$'\,]_t^o\,]_i^o\,]_b[\ \,]_c^o \cdots [\ \,]_z^o\,]_o^o$$

Consider a P system corresponding to the given dictionary:

$$\begin{array}{lcl} \Pi & = & \big(O,\Sigma,H,E,\mu,M_1,\cdots,M_p,R,i_0\big), \\ O & = & A_1\cup A_2\cup\{?,?',\$,\$',\$_1,\$_2,\mathtt{fail}\}\cup\{?_i \mid 1\le i\le 11\}\cup\{!_i \mid 1\le i\le 4\}, \\ \Sigma & = & A_1\cup A_2\cup\{?,?',!,\$,\$'\}, \\ H & = & A_1\cup\{0\},\ E=\{0,+,-\},\ i_0=1, \\ \mu & \text{and} & \text{sets } M_i,\ 1\le i\le p, \text{ are defined as described above.} \end{array}$$

So only the rules and input semantics still have to be defined.

3.1 Dictionary search

To translate a word u, input the string $?u?'$ in region 1. Consider the following rules.

S1 $?_l[\ ]_l^0 \to [\ ?\ ]_l^0,\ l\in A_1$

Propagation of the input into the membrane structure, reaching the location corresponding to the input word.

S2 $[\ ??'\ ]_l^0 \to [\ ]_l^-\emptyset,\ l\in A_1$

Marking the region corresponding to the source word.

S3 $[\ \$\to \$_1\|\$_2\ ]_l^-,\ l\in A_1$

Replicating the translation.

S4 $[\ \$_2\ ]_l^e \to [\ ]_l^0\$_2,\ l\in H,\ e\in\{-,0\}$

Sending one copy of the translation to the environment.

S5 $[\ \$_1\to\$\ ]_l^0,\ l\in A_1$

Keeping the other copy in the dictionary.

The system will send the translation of u in the environment. This is a simple example illustrating search. If the source word is not in the dictionary, the system will be blocked without giving an answer. The following subsection shows a solution to this problem.

3.2 Search with fail

The set of rules below is considerably more involved than the previous one. However, it handles 3 cases: a) the target word is found, b) the target word is missing in the target location, c) the target location is unreachable.

F1 $[\ ?\to ?_1\|?_2\ ]_0^0$

Replicate the input.

F2 $[\ ?_2\to ?_3\ ]_0^0$

Delay the second copy of the input for one step.

F3 $?_1l[\ ]_l^0 \to [\ ?_1\ ]_l^+,\ l\in A_1$

Propagation of the first copy towards the target location, changing the polarization of the entered membrane to $+$.

F4 $?_3 l[\ ]_l^+ \to [\ ?_3\ ]_l^{\mathrm{o}}, l \in A_1$

Propagation of the second copy towards the target location, restoring the polarization of the entered membrane.

F5 $[\ ?_1 l \to [\ ?_4\ ]_l^-\ ]_k^{\mathrm{o}}, l,k \in A_1$

If a membrane corresponding to some symbol of the source word is missing, then the first copy of the input remains in the same membrane, while the second copy of the input restores its polarization. Creating a membrane to handle the failure.

F6 $[\ ?_1 ?' \to ?_7\ ]_l^{\mathrm{o}}, l \in A_1$

Target location found, marking the first input copy.

F7 $[\ ?_7\ ]_l^{\mathrm{o}} \to [\ ]_l^- \emptyset, l \in A_1$

Marking the target location.

In either case, some membrane has polarization $-$. It remains to send the answer out, or fail if it is absent. The membrane should be deleted in the fail case.

F8 $[\ \$ \to \$_1 || \$_2\ ]_l^-, l \in A_1$

Replicating the translation.

F9 $[\ \$_2\ ]_l^e \to [\ ]_l^{\mathrm{o}} \$_2, l \in H, e \in \{\mathrm{o}, -\}$

Sending one copy of the translation out.

F10 $[\ \$_1 \to \$\ ]_l^{\mathrm{o}}, l \in A_1$

Keeping the other copy in the dictionary.

F11 $[\ ?_3 \to ?_5\ ]_l^-, l \in A_1$

The second copy of input will check if the translation is available in the current region.

F12 $?_3 l[\ ]_l^- \to [\ ?_5\ ]_l^-, l \in A_1$

The second copy of input enters the auxiliary membrane with polarization $-$.

By now the second copy of the input is in the region corresponding to either the search word, or to its maximal prefix plus one letter (auxiliary one).

F13 $[\ ?_5 \to ?_6\ ]_l^-, l \in A_1$

It waits for one step.

F14 $[\ ?_6 \to \emptyset\ ]_l^{\mathrm{o}}, l \in A_1$

If the target word has been found, the second copy of the input is erased.

F15 $[\ ?_6\ ]_l^- \to [\ ]_l^{\mathrm{o}} ?_8, l \in A_1$

If not, the search fails.

F16 $[\ ?_8\ ]_l^{\mathrm{o}} \to [\ ]_l^{\mathrm{o}} ?_8, l \in A_1$

Sending the fail notification to the skin.

F17 $[\ ?_8 l \to ?_8\ ]^{\mathrm{o}}_{\mathrm{o}}$

Erasing the remaining part of the source word.

F18 $[\ ?_8 ?'\ ]^{\mathrm{o}}_{\mathrm{o}} \to [\ ]^{\mathrm{o}}_{\mathrm{o}}\texttt{fail}$

Answering fail.

F19 $[\ ?_4 \to ?_9\ ]^{-}_{l}$, $l \in A_1$

F20 $[\ ?_9 \to ?_{10}\ ]^{-}_{l}$, $l \in A_1$

F21 $[\ ?_{10} \to ?_{11}\ ]^{-}_{l}$, $l \in A_1$

If the target location was not found, the first input copy waits for 3 steps while the membrane with polarization $-$ handles the second input copy.

F22 $[\ ?_{11}\ ]^{\mathrm{o}}_{l} \to \emptyset$, $l \in A_1$

Erasing the auxiliary membrane.

3.3 Dictionary update

To add an entry $u \longrightarrow v$ to the dictionary, input the string $!u\$v\$'$ in region 1. Consider the following rules.

U1 $[\ ! \to !_1 || !_2\ ]^{\mathrm{o}}_{\mathrm{o}}$

Replicate the input.

U2 $[\ !_2 \to !_3\ ]^{\mathrm{o}}_{\mathrm{o}}$

Delay the second copy of the input for one step.

U3 $!_1 l[\ ]^{\mathrm{o}}_{l} \to [\ !_1\ ]^{+}_{l}$, $l \in A_1$

Propagation of the first copy towards the target location, changing the polarization of the entered membrane to $+$.

U4 $!_3 l[\ ]^{+}_{l} \to [\ !_3\ ]^{\mathrm{o}}_{l}$, $l \in A_1$

Propagation of the second copy towards the target location, restoring the polarization of the entered membrane.

U5 $[\ !_1 \to !_4\ ]^{\mathrm{o}}_{l}$, $l \in A_1$

If a membrane corresponding to some symbol of the source word is missing, then the first copy of the input remains in the same membrane, while the second copy of the input restores its polarization. Marking the fist copy of the input for creation of missing membranes.

U6 $[\ !_4 l \to [\ !_4\ ]^{+}_{l}\ ]^{\mathrm{o}}_{k}$, $l,k \in A_1$

Creating missing membranes.

U7 $[\ !_4\$ \to \$\ ]^{\mathrm{o}}_{l}$, $l \in A_1$

Releasing the target word in the corresponding location.

U8 $[\ !_3\$ \to \emptyset\ ]^{\mathrm{o}}_{l}, l \in A_1$

Erasing the second copy of the input.

We underline that the constructions presented above also hold in a more general case, i.e., when the dictionary is a multi-valued function. Indeed, multiple translations can be added to the dictionary as multiple strings in the region associated to the input word. The search for a word with multiple translations will lead to all translations sent to the environment. The price to pay is that the construction is no longer deterministic, since the order of application of rules S4 or F9 to different translations is arbitrary. Nevertheless, the constructions remain "deterministic modulo the order in which the translations are sent out". All constructions work in linear time with respect to the length of the input. The parallelism is vital for checking for the absence of a needed submembrane, or for checking for the absence of a translation of a given word; sending multiple translation results out is also parallel.

4 Discussion

In this paper we presented the linear-time algorithms of searching in a dictionary and updating it implemented as membrane systems. We underline that the systems are constructed as reusable modules, so they are suitable for using as sub-algorithms for solving more complicated problems.

The scope of handling dictionaries is not limited to the dictionaries in the classical sense. Understanding a dictionary as introduced in Section 3, i.e., a string-valued function defined on a finite set of strings, leads to direct applicability of the proposed methods to handle alphabets, lexicons, thesauruses, dictionaries of exceptions, and even databases. At last, it is natural to consider these algorithms together with morphological analyzer and morphological synthesizer.

Acknowledgments All authors gratefully acknowledge the support by the Science and Technology Center in Ukraine, project 4032. 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. Yurii Rogozhin gratefully acknowledges the support of the European Commission, project MolCIP, MIF1-CT-2006-021666.

Bibliography

[1] G. Ciobanu, G. Păun, M.J. Pérez-Jiménez Eds., Applications of Membrane Computing, Springer-Verlag, 2006.

[2] H. Kitano, Challenges of Massive Parallelism, Proceedings of the *13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993, vol. 1, 813–834.

[3] Gh. Păun, Computing with Membranes, *Journal of Computer and System Sciences* **61**(1), 2000, 108–143.

[4] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, 2002.

[5] E. Sumita, K. Oi, O. Furuse, H. Iida, T. Higuchi, N. Takahashi, H. Kitano, Example-Based Machine Translation on Massively Parallel Processors, Proceedings of the *13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993, vol. 2, 1283–1289.

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

Artiom Alhazov (born on October 11, 1979), graduated in Mathematics and Computer Science (The State University of Moldova, Chişinău, Moldova) and received Ph.D. in Languages and Information Systems (Rovira i Virgili University, Tarragona, Spain). A researcher at the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova. He completed a postdoc in Åbo Akademi University, Turku, Finland, and currently has a postdoc in Hiroshima University, Higashi-Hiroshima, Japan. His main research interests are theoretical computer science, formal language theory, parallel distributed computational models, and in particular the descriptional complexity of P systems with weak forms of interaction. He published over 90 research papers (collaborating with more than 30 researchers from many countries in Europe and Asia). He won numerous prizes for programming in school and university years, and the National Youth Prize in Science, Technics, Literature and Arts in 2006 for a collection of research works.

Svetlana Cojocaru (born on July 26, 1952), graduated in Mathematics (The State University of Moldova, Chişinău, Moldova, 1974), received Ph.D. in Computer Science (Institute of Cybernetics, Ukrainian Academy of Sciences, Kiev, 1982) and doctor in habilitation in Computer Science (Institute of Mathematics and Computer Science, Academy of Sciences of Moldova, 2007). A deputy director at the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova. Her main research interests are formal languages and grammars, natural language processing, computer algebra, molecular computing. She published over 120 research papers.

Ludmila Malahova (born on July 22, 1947), graduated in Computer Science (The State University of Moldova, Chişinău, Moldova, 1970). A researcher at the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova. She has a significant experience in computer science including computer graphics, formal languages, computer algebra, natural language processing, and molecular computing with more than 80 papers published in international journals, books and conference proceedings.

Yurii Rogozhin (born on November 13, 1949), graduated in Mathematics (The Kuban State University, Krasnodar, Russia, 1971), received Ph.D. in Mathematical Cybernetics (Computer Center of the Russian Academy of Sciences, Moscow, 1981) and doctor in habilitation in Computer Science (Moscow State University, Russia, Department of Computational Mathematics and Cybernetics, 1999). A principal researcher at the Institute of Mathematica and Computer Science of the Academy of Sciences of Moldova Republic and Marie Curie IIF researcher at Rovira i Virgili University, Research Group on Mathematical Linguistics, Tarragona, Spain. His main research interests are mathematics, theoretical computer science, formal language theory and its applications, natural (biomolecular) computing and nanotechnology. He published over 110 research papers (collaborating with more than 45 researches from many countries in Europe).