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

Artiom Alhazov[2,1], Svetlana Cojocaru[1], Ludmila Malahova[1],
Yurii Rogozhin[3,1]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova, Academiei 5, Chişinău MD-2028 Moldova
`{artiom,sveta,mal,rogozhin}@math.md`
[2] IEC, Department of Information Engineering, Graduate School of Engineering
Hiroshima University, Higashi-Hiroshima 739-8527 Japan
[3] Rovira i Virgili University, Research Group on Mathematical Linguistics
Pl. Imperial Tàrraco 1, Tarragona 43005 Spain

Summary. Membrane computing is a formal framework of distributed parallel computing. In this paper we implement working with the prefix tree by P systems with strings and active membranes.

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 solution is that based on applying parallelism; it has started to be promoted already in 1970s. For instance, the possibilities of applying massive parallelism in Machine Translation are considered in [4, 1]. We mention that 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 applying 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 there are handling of the dictionaries, e.g., dictionary lookup and dictionary completion. Exactly these problems constitute the subject of the present paper. In our approach we speak about dictionary represented by a prefix tree.

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 started by Gh. Păun in [2]. 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 [3] for an overview of the domain and [5] for the comprehensive bibliography.

2.1 Computing by P systems

Let O be a finite set of elements called symbols; the 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 whose elements are called objects.*
- *Σ is an input alphabet.*
- *H is an alphabet whose elements are called labels.*
- *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, $1 \le i \le p$.*
- *R is a finite set of rules defining the behavior of objects from O and membranes labeled by elements of H.*
- *i_0 identifies the input region.*

A configuration of a P system is its "snapshot", i.e., the current membrane structure and the multisets of strings of objects present in regions of the system. While 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, 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

$$(\mu, M_1, \cdots, M_{i_0-1}, M_{i_0} \cup M, M_{i_0+1}, \cdots, M_p).$$

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 can consider consecutive symbols in a string (otherwise, the time complexity would be much higher).
- 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, in the sense that 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 $h \in H, e \in E, a, b, c \in O^*$
(like the previous case, 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; the object can be modified during this process, as well as 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; the object can be modified during this process; 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 and 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).

The rules can only be applied simultaneously if they involve different objects and membranes (we repeat that rules of type (a) are not considered to involve a membrane), and such parallelism is maximal if no further rules are applicable to objects and membranes that were not involved.

3 Dictionary

Dictionary search represents computing a string-valued function

$$\{u_i \longrightarrow v_i \mid 1 \leq i \leq d\}$$

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 $\{u_i \mid 1 \leq i \leq d\}$, with strings $\$v_i\$'$ in regions corresponding to the nodes associated to u_i. 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

$$[\ [\ \ ]^0_a [\ [\ [\ \$flying\$'\ ]^0_t\ ]^0_a [\ [\ \$stored\$'\ ]^0_t\ ]^0_i\ ]_b [\ \ ]^0_c \cdots [\ \ ]^0_z\ ]^0_0$$

Let A_1, A_2 be the alphabets of the source and target languages, respectively. Consider a P system corresponding to the given dictionary.

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

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

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

F4 $?_3 l[\ ]_l^+ \to [\ ?_3\ ]_l^0,\ l \in A_1$

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

F5 $[\ ?_1 l \rightarrow [\ ?_4\ ]_l^-\ ]_k^0,\ l,k \in A_1$

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 $[\ ?_1 ?' \rightarrow ?_7\ ]_l^0,\ l \in A_1$

Target location found, marking the first input copy.

F7 $[\ ?_7\ ]_l^0 \rightarrow [\ \ ]_l^- \emptyset,\ l \in A_1$

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

Replicating the translation.

F9 $[\ \$_2\ ]_l^e \rightarrow [\ \ ]_l^0 \$_2,\ l \in H,\ e \in \{0, -\}$

Sending one copy of the translation out.

F10 $[\ \$_1 \rightarrow \$\ ]_l^0,\ l \in A_1$

Keeping the other copy in the dictionary.

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

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

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

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

It waits for one step.

F14 $[\ ?_6 \rightarrow \emptyset\ ]_l^0,\ l \in A_1$

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

F15 $[\ ?_6\ ]_l^- \rightarrow [\ \ ]_l^0 ?_8,\ l \in A_1$

If not, the search fails.

F16 $[\ ?_8\ ]_l^0 \rightarrow [\ \ ]_l^0 ?_8,\ l \in A_1$

Sending the fail notification to the skin.

F17 $[\ ?_8 l \rightarrow ?_8\ ]_0^0$

Erasing the remaining part of the source word.

F18 $[\ ?_8?'\ ]_l^0 \to [\ ]_l^0$`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 $[\ ?_{11}\ ]_l^0 \to \emptyset$, $l \in A_1$

Erasing the auxiliary membrane.

3.3 Dictionary update

To add a pair of words $u \longrightarrow v$ to the dictionary, input the string $!u\$v\$'$ in region 1. Consider the following rules.

U1 $[\ ! \to !_1 || !_2\ ]_0^0$

Replicate the input.

U2 $[\ !_2 \to !_3\ ]_0^0$

Delay the second copy of the input for one step.

U3 $!_1 l [\ ]_l^0 \to [\ !_1\ ]_l^+$, $l \in A_1$

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

U4 $!_3 l [\ ]_l^+ \to [\ !_3\ ]_l^0$, $l \in A_1$

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

U5 $[\ !_1 \to !_4\ ]_l^0$, $l \in A_1$

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 $[\ !_4 l \to [\ !_4\ ]_l^+\ ]_k^0$, $l, k \in A_1$

Creating missing membranes.

U7 $[\ !_4\$ \to \$\ ]_l^0$, $l \in A_1$

Releasing the target word in the corresponding location.

U8 $[\ !_3\$ \to \emptyset\ ]_l^0,\ l \in A_1$

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".

4 Discussion

In this paper we presented the algorithms of searching in a dictionary and completing 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, thesaura, dictionaries of exceptions, and even databases.

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.

References

1. H. Kitano: Challenges of massive parallelism. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993, vol. 1, 813–834.
2. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
3. Gh. Păun: *Membrane Computing. An Introduction.* Springer-Verlag, 2002.
4. 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.
5. P systems webpage. `http://ppage.psystems.eu/`.