Decidability of the termination problem for sequential P systems with active membranes

Michal Kováč

Faculty of Mathematics, Physics and Informatics, Comenius University

Abstract. Abstract

1 Introduction

Membrane systems (P systems) were introduced by Păun (see [1]) as distributed parallel computing devices inspired by the structure and functionality of cells. One of the objectives is to relax the condition of using the rules in a maximally parallel way in order to find more realistic P systems from a biological point of view. In sequential systems, only one rewriting rule is used in each step of computation.

2 Preliminaries

Here we recall several notions from the classical theory of formal languages.

An **alphabet** is a finite nonempty set of symbols. Usually it is denoted by Σ or V. A **string** over an alphabet is a finite sequence of symbols from alphabet. We denote by V^* the set of all strings over an alphabet V. By $V^+ = V^* - \{\varepsilon\}$ we denote the set of all nonempty strings over V. A **language** over the alphabet V is any subset of V^*.

The number of occurrences of a given symbol $a \in V$ in the string $w \in V^*$ is denoted by $|w|_a$. $\Psi_V(w) = (|w|_{a_1}, |w|_{a_2}, \dots, |w|_{a_n})$ is called a Parikh vector associated with the string $w \in V^*$, where $V = \{a_1, a_2, \dots a_n\}$. For a language $L \subseteq V^*$, $\Psi_V(L) = \{\Psi_V(w) | w \in L\}$ is the Parikh mapping associated with L. If FL is a family of languages, PsFL is denoted the family of Parikh images of languages in FL.

A multiset over a set X is a mapping $M : X \to \mathbb{N}$. We denote by $M(x), x \in X$ the multiplicity of x in the multiset M. The **support** of a multiset M is the set $supp(M) = \{x \in X | M(x) \geq 1\}$. It is the set of items with at least one occurrence. A multiset is **empty** when its support is empty. A multiset M with finite support $X = \{x_1, x_2, \dots, x_n\}$ can be represented by the string $x_1^{M(x_1)} x_2^{M(x_2)} \dots x_n^{M(x_n)}$. As elements of a multiset can also be strings, we separate them with the pipe symbol, e.g. *element*|*element*|*other_element*. We say that multiset M_1 is included in multiset M_2 if $\forall x \in X : M_1(x) \leq M_2(x)$. We denote it by $M_1 \subseteq M_2$. If $M_1 \subseteq M_2$, the **difference** of two multisets $M_2 - M_1$ is defined as a multiset where $\forall x \in X : (M_2 - M_1)(x) = \max(M_2(x) - M_1(x), 0)$. The **union** of two multisets $M_1 \cup M_2$ is a multiset where $\forall x \in X : (M_1 \cup M_2)(x) = M_1(x) + M_2(x)$. The product of multiset M with natural number $n \in \mathbb{N}$ is a multiset where $\forall x \in X : (n \cdot M)(x) = n \cdot M(x)$.

3 Active P systems

The fundamental ingredient of a P system is the **membrane structure** (see [2]). It is a hierarchically arranged set of membranes, all contained in the **skin membrane**. Each membrane determines a compartment, also called region, which is the space delimited from above by it and from below by the membranes placed directly inside, if any exists. Clearly, the correspondence membrane – region is one-to-one, that is why we sometimes use interchangeably these terms.

Consider a finite set of symbols $V = \{a_1, a_2, \ldots, a_n\}$. An arbitrary multiset rewriting rule is a pair (u, v) of multisets over the set V where u is not empty. Such a rule is typically written as $u \to v$. For a multiset rewriting rule $r : u \to v$, let $left(r) = u$ and $right(r) = v$. Let w be a multiset of symbols over V and let $R = \{r_1, r_2, \ldots, r_k\}$ be a set of multiset rewriting rules such that $r_i = u_i \to v_i$ with multisets u_i, v_i over V. Denote by $R_w^{ap} \subseteq R$ the **set of applicable multiset rewriting rules** to w, that is, $R_w^{ap} = \{r \in R | left(r) \subseteq w\}$.

P system with active objects is a tuple $(V, \mu, R_1, R_2, \ldots, R_m)$, where:

- V is the alphabet of symbols,
- μ is initial membrane structure containing multisets of objects from V, defined recursively as:

$$[_1 M]_1$$
$$M ::= \varepsilon | Ma | M[_j M]_j, \text{ where } a \in V \text{ and } 1 \leq j \leq m$$

- $R_1, R_2, \ldots R_m$ are finite sets of rewriting rules associated with the membranes labeled $1, 2, \ldots, m$ and can be of forms:
  - $u \to v$, where $u \in V^+$, $v \in (V \times \{\cdot, \uparrow, \downarrow_j\})$ and $1 \leq j \leq m$
  - $u \to v\delta$, where $u \in V^+$, $v \in (V \times \{\cdot, \uparrow, \downarrow_j\})$ and $1 \leq j \leq m$
  - $u \to [_j w]_j$, where $w \in V^*$ and $1 \leq j \leq m$

Each rewriting rule may specify for each symbol on the right side, whether it stays in the current region (we will omit the symbol $\cdot$), moves through the membrane to the parent region ($\uparrow$) or to a specific child region ($\downarrow_m$, where m is a label of a membrane). We denote these transfers with arrows immediately after the symbol. An example of such rule is the following: $abb \to ab \downarrow_2 c \uparrow c$.

Symbol δ at the end of the rule means that after application of the rule, the membrane is dissolved and it's contents (objects, child membranes) are propagated to the parent membrane.

A **configuration** of a P system is represented by its membrane structure and the multisets of objects in the regions.

A **computation step** of P system is a relation $\Rightarrow$ on the set of configurations such that $C_1 \Rightarrow C_2$ iff there is an applicable rule in a membrane in C_1 such that applying that rule would result in C_2.

A **computation** of a P system consists of a sequence of steps. The step S_i is applied to result of previous step S_{i-1}. So when $S_i = (C_j, C_{j+1})$, $S_{i-1} = (C_{j-1}, C_j)$.

There are two possible ways of assigning a result of a computation:

1. By considering the multiplicity of objects present in a designated membrane in a halting configuration. In this case we obtain a vector of natural numbers. We can also represent this vector as a multiset of objects or as Parikh image of a language.
2. By concatenating the symbols which leave the system, in the order they are sent out of the skin membrane (if several symbols are expelled at the same time, then any ordering of them is considered). In this case we generate a language.

The result of a computation is clearly only one multiset or a string, but for one initial configuration there can be multiple possible computations. It follows from the fact that there exist more than one maximal multiset of rules that can be applied in each step.

3.1 Register machines

As a referential universal language acceptor we will use Minsky's register machine. Such a machine runs a program consisting of numbered instructions of several simple types.

Definition 1. *A n-**register machine** is a tuple $M = (n, P, i, h)$, where:*

- *n is the number of registers,*
- *P is a set of labeled instructions of the form $j : (op(r), k, l)$, where $op(r)$ is an operation on register r of M, and j, k, l are labels from the set $Lab(M)$ (which numbers the instructions in a one-to-one manner),*
- *i is the initial label, and*
- *h is the final label.*

The machine is capable of the following instructions:

- $(add(r), k, l)$: Add one to the contents of register r and proceed to instruction k or to instruction l; in the deterministic variants usually considered in the literature we demand $k = l$.
- $(sub(r), k, l)$: If register r is not empty, then subtract one from its contents and go to instruction k, otherwise proceed to instruction l.
- *halt* : This instruction stops the machine. This additional instruction can only be assigned to the final label h.

A deterministic m-register machine can analyze an input $(n_1, \dots, n_m) \in N_0^m$ in registers 1 to m, which is recognized if the register machine finally stops by the halt instruction with all its registers being empty (this last requirement is not necessary). If the machine does not halt, the analysis was not successful.

4 Conclusion

We have studied ...

References

1. Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1) (2000) 108 – 143
2. Păun, G.: Introduction to membrane computing. In Ciobanu, G., Păun, G., Pérez-Jiménez, M., eds.: Applications of Membrane Computing. Natural Computing Series. Springer Berlin Heidelberg (2006) 1–42