Computer Science Journal of Moldova, vol.**20**, no.3(60), 2012

P systems based on tag operations

Yurii Rogozhin Sergey Verlan

Abstract

In this article we introduce P systems using Post's tag operation on strings. We show that the computational completeness can be achieved even if the deletion length is equal to one.

1 Introduction

The tag operation was invented by E. Post during his Procter fellowship at Princeton during the academic year 1920-21 [12, 13]. This operation deletes first n letters of a word and appends an appendant depending on the first deleted letter. Computational devices based on this operation, *the tag systems*, are one of the simplest examples of universal devices [8, 3]. The number of deleted symbols, *the deletion number*, permits to establish a frontier between decidability and undecidability – if it is equal to two, then the corresponding class is undecidable, while if it is equal to one, then the corresponding class is decidable. There exist other interesting properties of tag systems, we refer to [7] for a review on the recent results in this field.

P systems [10, 11] are distributed computational devices inspired from the structure and the functioning of a living cell. The cell is considered as a set of compartments (membranes) nested one in another and which contain objects and evolution rules. The base model does not specify neither the nature of these objects, nor the nature of rules. Numerous variants specify these two parameters by obtaining a lot of different models of computing, see [15] for a comprehensive bibliography.

In the case of P systems with tag operations the basic objects are strings and the operations in membranes are tag operations. In a formal way, an n-tag P systems can be considered like a graph, whose nodes contain sets of strings and sets of tag rules with the deletion number n. Every rule permits to perform a tag operation and to send the result to some other node. Such an approach is close to the idea of graph-controlled or programmed grammars, where a similar control mechanism is used, but for rewriting rules. We show that using P systems permits to strictly increase the power of the tag operation and to achieve the universality with the deletion number equal to one.

2 Definitions

In this section we recall some very basic notions and notations we use throughout the paper. We assume the reader to be familiar with the basics of formal language theory. For more details, we refer to [14].

A *tag system* of degree $m > 0$, see [3] and [9], is the triplet $T = (m, V, P)$, where $V = \{a_1, \dots, a_{n+1}\}$ is an alphabet and where P is a set of productions (tag operations) of form $a_i \rightarrow P_i, 1 \leq i \leq n$, $P_i \in V^*$. We remark that for every a_i, $1 \leq i \leq n$, there is exactly one production in P. The value m is also called the deletion number of T. The symbol a_{n+1} is called the halting symbol. A configuration of the system T is a word w. The application of the tag operation permits to pass from a configuration $w = a_{i_1} \dots a_{i_m} w'$ to the next configuration z by erasing the first m symbols of w and by adding P_{i_1} to the end of the word: $w \Longrightarrow z$, if $z = w' P_{i_1}$.

The computation of T over the word $x \in V^*$ is a sequence of configurations $x \Longrightarrow \dots \Longrightarrow y$, where either $y = a_{n+1} a_{i_1} \dots a_{i_{m-1}} y'$, or $y' = y$ and $|y'| < m$. In this case we say that T halts on x and that y' is the result of the computation of T over x. We say that T recognizes the language L if there exist a recursive coding ϕ such that for all $x \in L$, T halts on $\phi(x)$, and T halts only on words from $\phi(L)$.

We note that tag systems of degree 2 are able to recognize the family of recursively enumerable languages [3, 9]. Moreover, the construction in [3] has non-empty productions and halts only by reaching the symbol a_{n+1} in the first position. It is also known that tag systems of degree 1 are decidable [6, 16]. It thus follows that the deletion number m is one decidability criterion [5] for tag systems with $m = 2$ as the frontier value.

Now we introduce the notion of the circular Post machine (CPM).

Definition 1 *A* circular Post machine *(of type 0) is a tuple* $(\Sigma, Q, \mathbf{q}_1, \mathbf{q}_f, R)$ *with a finite alphabet* Σ *where* $0 \in \Sigma$ *is the blank, a finite set of states* Q*, the initial state* $\mathbf{q}_1 \in Q$*, the final state* $\mathbf{q}_f \in Q$*, and a finite set of instructions* R *with all instructions having one of the forms* $\mathbf{p}x \to \mathbf{q}$ *(erasing the symbol read by deleting a symbol),* $\mathbf{p}x \to y\mathbf{q}$ *(overwriting and moving to the right),* $\mathbf{p}0 \to y\mathbf{q}0$ *(overwriting and inserting a blank symbol), where* $x, y \in \Sigma$ *and* $\mathbf{p}, \mathbf{q} \in Q$*,* $\mathbf{p} \neq \mathbf{q}_f$*.*

We also refer to all instructions with $\mathbf{q}_f$ in the right hand side as halt instructions. The storage of this machine is a circular tape, the read and write head moves only in one direction (to the right), and with the possibility to delete a cell or to create and insert a new cell with a blank.

Notice that a circular tape can be thought of as a finite string of symbols (from the one following the state to the one preceding the state in the circular representation). In this way, CPM0 is a finite-state machine, which reads the leftmost symbol of the string, possibly consuming it, and uses the symbol+state information to change the state, possibly writing a symbol on the right.

There are several other variants of CPM [4, 1] which differ in the way the lengthening instructions work. All these variants are computationally equivalent, although their descriptional complexity can be different.

Now we define P systems that use the tag operation.

An *n-tag P system* is the construct

$\Pi = (O, T, \mu, M_1, \cdots, M_n, R_1, \cdots, R_n)$, where

- O is a finite alphabet,
- $T \subseteq O$ is the terminal alphabet,

- μ is the membrane (tree) structure of the system which has n membranes (nodes) and it can be represented by a word over the alphabet of correctly nested marked parentheses,
- M_i, for each $1 \leq i \leq n$ is a finite language associated to the membrane i,
- R_i, for each $1 \leq i \leq n$ is a set of rules associated to membrane i, of the following forms: $a \rightarrow P_a; tar$, $a \in O$ where $a \rightarrow P_a$ is a tag rule and tar is the *target indicator* from the set $\{here, in_j, out \mid 1 \leq j \leq n\}$, where j is a label of the immediately inner membrane of membrane i.

An n-tuple $(N_1, \cdots, N_n)$ of finite languages over O is called a configuration of Π. The transition between the configurations consists of applying the tag rules (with the deletion length n) in parallel to all possible strings, non-deterministically, and following the target indications associated with the rules.

More specifically, if $w = aa_2 \dots a_n w' \in N_i$ and $r = a \rightarrow P_a; tar$ then the word $w'P_a$ will go to the region indicated by tar. If $tar = here$, then the string remains in N_i, if $tar = out$, then the string is moved to the region immediately outside the membrane i (maybe, in this way the string leaves the system), if $tar = in_j, j = 1, ..., n$, then the string is moved to the immediately below j-th region.

A sequence of transitions between configurations of a given insertion-deletion P system Π, starting from the initial configuration $(M_1, \dots, M_n)$, is called a computation with respect to Π. The result of a computation consists of all strings over T which are sent out of the system at any time during the computation. We denote by $L(\Pi)$ the language of all strings of this type. We say that $L(\Pi)$ is generated by Π.

We denote by $ELSP_k(n - tag)$ the family of languages $L(\Pi)$ generated by n-tag P systems with $k \geq 1$ membranes.

3 Results

Theorem 1 *Any CPM0 M can be simulated by a 1-tag P system.*

Proof. Consider a CPM0 $M = (\Sigma, Q, q_1, q_f, R)$ with symbols $\Sigma = \{a_j \mid 0 \leq j \leq n\}$, where $a_0 = 0$ is the blank symbol, and states $Q = \{q_i \mid 1 \leq i \leq f\}$, where q_1 is the initial state and the only terminal state is $q_f \in Q$; let $Q' = Q \setminus \{q_f\}$.

Consider the following 1-tag P system

$$\Pi = (V, \Sigma, \mu, M_{m_s}, \dots, M_{m_f}, R_{i_s}, \dots, R_{i_f}) :$$

$$\begin{aligned} V &= \Sigma \cup Q, \\ \mu &= [\prod_{q_i a_j \in Q \times \Sigma} \Big([\ ]_{m_{ij}} \Big)]_{m_s}\ , \\ M_i &= \emptyset,\ i \neq m_s, \text{ and the rules are given and explained below.} \end{aligned}$$

Hence the membrane structure of Π consists of the skin membrane m_s and inner membranes $m_{ij},\ 1 \leq i \leq f, 0 \leq j \leq n$. The set of rules is defined as follows:

$$\begin{aligned} R_{m_s} &= \{1.ij : q_i \to \varepsilon; m_{ij} \mid 1 \leq i \leq f-1,\ 0 \leq j \leq n\} \\ &\cup \{2.j : a_j \to a_j; here \mid a_j \in \Sigma\} \\ &\cup \{3 : q_f \to \varepsilon; out\}, \\ R_{m_{ij}} &= \{4.ij : a_j \to a_k q_l; out \mid q_i a_j \to a_k q_l \in R, j > 0\} \\ &\cup \{5.ij : a_j \to q_l; out \mid q_i a_j \to q_l \in R, j > 0\} \\ &\cup \{6.i : a_0 \to a_k q_l a_0; out \mid q_i a_0 \to a_k q_l a_0 \in R\}. \end{aligned}$$

A configuration $v = q_i a_j W$ of M describes that M in state $q_i \in Q$ considers symbol $a_j \in \Sigma$ to the left of $W \in \Sigma^*$. This configuration is encoded by the string v in the skin membrane m_s of Π.

The machine M starts a computation from a configuration $q_1 a_j W$ and Π starts computation from the corresponding string $q_1 a_j W$ in membrane m_s (other regions of Π are empty). We shall show now how the rules of M are simulated in Π.

Consider rule $\mathbf{q_i a_j \to a_k q_l \in R}, q_i \in Q', q_l \in Q, a_j, a_k \in \Sigma$ of M. It is simulated in Π as follows.

Let $q_ia_jW \overset{q_ia_j\to a_kq_l}{\Longrightarrow} q_lW$ be a computation step in M, i.e., rule $q_ia_j \to a_kq_l$ is applied to configuration q_ia_jW yielding q_lWa_k ($W \in \Sigma^*$).

This rule is simulated in Π as follows. One of rules $1.ip$ is non-deterministically applied to string q_ia_jW and the resulting string a_jW moves to region m_{ip}. We denote this action as follows:

$$(m_s, q_ia_jW) \overset{1.ip}{\Longrightarrow} (m_{ip}, a_jW).$$

If $p \neq j$, then the corresponding string cannot evolve anymore as there is no applicable rule in membrane m_{ip}. If $p = j$, then the following evolution is possible yielding Wa_kq_l in the skin membrane:

$$(m_{ij}, a_jW) \overset{4.ij}{\Longrightarrow} (m_s, Wa_kq_l).$$

Next, the only possibility to continue is to apply the group of rules $2.j$ until string q_lWa_k is obtained:

$$(m_s, Wa_kq_l) \overset{2.j_1}{\Longrightarrow} \dots \overset{2.j_t}{\Longrightarrow} (m_sq_lWa_k).$$

Thus we showed that Π correctly simulates rule $q_ia_j \to a_kq_l$ of M.

It is not difficult to see that rules of type $\mathbf{q_ia_j \to q_l}$, $q_i \in Q', q_l \in Q, a_j \in \Sigma$, resp. $\mathbf{q_ia_0 \to a_kq_la_0}$, $q_i \in Q', q_l \in Q, a_j \in \Sigma$, can be simulated in a similar manner replacing $4.ij$ by $5.ij$, resp. $6.ij$.

We observe that for a string that reached a halting configuration q_fW in M, only rule 3 is applicable on the corresponding string q_fW of Π. This leads to the word W that is sent out of the system.

Hence we obtain that for any transition $w \Longrightarrow w'$ in M there is a unique sequence of transitions $(m_s, w) \Longrightarrow (m_{ij}, w_1)\dots \Longrightarrow (m_s, w_k) \Longrightarrow (m_s, w')$ in Π, for some $w_j \in O^*$ and $k > 0$. □

Corollary 1 *There exists a universal 1-tag P system with 73 instructions.*

Proof. Consider the universal CPM0 from [2]. It has 6 states and 6 symbols. By applying Theorem 1 to this machine we obtain a universal 1-tag P system with 73 rules. □

4 Conclusion

In this article we considered the tag operation in the context of P systems. The obtained variant is universal even with the deletion number equal to one. Moreover, the obtained system has 73 instructions while best actually known constructions for universal tag systems have around 480 [7]. An open problem is if this number can be decreased.

P systems framework for the tag operation can be considered as a particular variant of the graph-controlled derivation using the tag operation. We observe that the particular structure of the graph from Theorem 1 corresponds to a matrix control with the depth (size of the matrices) equal to two. Hence Corollary 1 also holds for matrix tag systems. It could be interesting to consider other control mechanisms like random-context control with the tag operation.

References

[1] A. Alhazov, A. Krassovitskiy, Yu.Rogozhin. *Circular Post Machines and P Systems with Exo-insertion and Deletion.* Lecture Notes in Computer Science, **7184** (2011), pp. 73–86.

[2] A. Alhazov, M. Kudlek, Yu. Rogozhin. *Nine Universal Circular Post Machines.* Computer Science Journal of Moldova, **10, no.3** (2002), pp. 247–262.

[3] J. Cocke, M. Minsky. *Universality of tag systems with p=2.* Journal of the ACM, **11**, 1, (1964), pp. 15–20.

[4] M. Kudlek, Yu. Rogozhin. *Small Universal Circular Post Machines.* Computer Science Journal of Moldova, **9(1)** (2001), pp. 34–52.

[5] M. Margenstern. *Frontier between decidability and undecidability: A survey,* Theoretical Computer Science, **231**(2) (2000), pp. 217–251.

[6] S. Maslov. *On E. L. Posts Tag problem.*, (In Russian) Trudy Matematicheskogo Instituta imeni V.A. Steklova (1964b), no. 72, pp. 5-56, English translation in: American Mathematical Society Translations Series 2, 97, pp. 1–14, 1971.

[7] L. De Mol. *On the complex behavior of simple tag systems – An experimental approach.* Theoretical Computer Science, **412**(1-2) (2011), pp. 97–112.

[8] M. Minsky. *Recursive unsolvability of Posts problem of tag and other topics in the theory of Turing machines*, Annals of Mathematics, **74** (1961), pp. 437–455.

[9] M. Minsky. *Computations: Finite and Infinite Machines.* Prentice Hall, Englewood Cliffts, NJ (1967).

[10] G. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

[11] G. Păun, G. Rozenberg, A. Salomaa (Eds.): *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.

[12] E. Post. *Formal reductions of the general combinatorial decision problem*, American Journal of Mathematics, **65**(2) (1943), pp. 197–215.

[13] E. Post. *Absolutely unsolvable problems and relatively undecidable propositions – account of an anticipation, The Undecidable.* In Martin Davis, ed., Basic papers on undecidable propositions, unsolvable problems and computable functions, Raven Press, 1965, pp. 340–433.

[14] G. Rozenberg, A. Salomaa. *Handbook of Formal Languages,* 3 volumes. Springer Verlag, Berlin, Heidelberg, New York (1997).

[15] The P systems Web page. `http://ppage.psystems.eu/`

[16] H. Wang. *Tag systems and lag systems,* Mathematische Annalen, **152** (1963a), pp. 65–74.

Yu. Rogozhin[1], S. Verlan[2,1],

Received July 9, 2012

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova

[2] LACL, Departement Informatique
UFR Sciences et Technologie
Universite Paris Est – Créteil Val de Marne
61, av. Géńeral de Gaulle
94010 Creteil, France

E–mails:
Dr.hab. Yurii Rogozhin: *rogozhin@math.md*,
Dr.hab. Sergey Verlan: *verlan@univ-paris12.fr*,