Generalized P Colony Automata and Their Relation to P automata [⋆] (Extended Abstract)

Kristóf Kántor, György Vaszil

Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
{kantor.kristof, vaszil.gyorgy}@inf.unideb.hu

1 Introduction

P colonies are variants of very simple membrane systems, which resemble the so-called colonies of grammars, see [7], that is, collections of very simple generative grammars which work in cooperation, and together they are able to generate fairly complicated languages.

P colonies also consist of a collection of very simple computing agents which interact in a shared environment, see [8, 9]. The agents and the environment described by multisets of objects which are processed by rules enabling the transformation of the objects and the exchange of objects between the colony members and the environment. The rules are grouped into programs, and a computation consists of a sequence of computational steps during which the colony members execute their programs in parallel, until the system reaches a halting configuration.

P colony automata, a variant of P colonies characterizing string languages were introduced in [2], the variants called generalized P colony automata were introduced in [5]. One of the motivations of introducing generalized P colony automata was to make the model resemble more to P automata, which was introduced in [4]. In the case of generalized P colony automata, the computation of the colony defines an accepted multiset sequence, which is turned into a set of accepted string by a non-erasing mapping (as in P automata).

2 P Automata and Generalized P Colony Automata

A *genPCol automaton* of capacity k and with n cells, $k, n \geq 1$, is a construct $\Pi = (V, e, w_E, (w_1, P_1), \ldots, (w_n, P_n), F)$ where

- V is the *alphabet* of the automaton, its elements are called *objects*;
- $e \in V$ is the *environmental object* of the automaton;

[⋆] Research supported in part by grant no. MAT120558 of the National Research, Development and Innovation Office, Hungary.

– $w_E \in (V - \{e\})^*$ is a string representing the multiset of objects different from e which is found in the environment initially;
– $(w_i, P_i), 1 \leq i \leq n$, specifies the i-th *cell* where w_i is a multiset over V, it determines the initial contents of the cell, and its cardinality $|w_i| = k$ is called the *capacity* of the system. P_i is a set of *programs*, each program is formed from k rules of the following types (where $a, b \in V$):
  - *tape rules* of the form $a \overset{T}{\rightarrow} b$, or $a \overset{T}{\leftrightarrow} b$, called rewriting tape rules and communication tape rules, respectively; or
  - *nontape rules* of the form $a \rightarrow b$, or $a \leftrightarrow b$, called rewriting (nontape) rules and communication (nontape) rules, respectively.

  A program is called a *tape program* if it contains at least one tape rule. (The names "tape" rule and "tape" program are motivated by the effect of the use of these rules/programs: as "ordinary" automata read symbols by processing an input tape, P colony automata read symbols by applying these rules/programs. See below for more details.)
– F is a set of *accepting configurations* of the automaton which we will specify in more detail below.

A genPCol automaton reads an input word during a computation. A part of the input (possibly consisting of more than one symbol) is read during each configuration change: the processed part of the input corresponds to the multiset of symbols introduced by the tape rules of the system.

A *configuration* of a genPCol automaton is an $(n+1)$-tuple $(u_E, u_1, \dots, u_n)$, where $u_E \in (V - \{e\})^*$ represents the multiset of objects different from e in the environment, and $u_i \in V^*,\ 1 \leq i \leq n$, represents the contents of the i-th cell. The *initial configuration* is given by $(w_E, w_1, \dots, w_n)$, the initial contents of the environment and the cells. The elements of the set F of *accepting configurations* are given as configurations of the form $(v_E, v_1, \dots, v_n)$, where

– $v_E \subseteq (V - \{e\})^*$ represents a multiset of objects different from e being in the environment, and each
– $v_i \in V^*,\ 1 \leq i \leq n$, is the contents of the i-th cell.

Instead of the different computational modes used in [2], in genPCol automata, we apply the programs in the maximally parallel way, that is, in each computational step, every component cell non-deterministically applies one of its applicable programs. Then we collect all the symbols that the tape rules "read" (these multisets are denoted by $read(p)$ for a program p in the definition below), this is the multiset read by the system in the given computational step. A successful computation defines in this way an accepted sequence of multisets: the sequence of multisets entering the system during the steps of the computation.

Let $\Pi = (V, e, w_E, (w_1, P_1), \dots, (w_n, P_n), F)$ be a genPCol automaton. The *set of input sequences accepted by* Π is defined as

$$\begin{aligned} A(\Pi) = \{u_1u_2\dots u_s \mid\ & u_i \in (V - \{e\})^*,\ 1 \leq i \leq s,\ \text{and there is a configuration} \\ & \text{sequence } c_0, \dots, c_s,\ \text{with } c_0 = (w_E, w_1, \dots, w_n),\ c_s \in F,\ \text{and} \\ & c_i \Longrightarrow c_{i+1} \text{ with } \bigcup_{p \in P_{c_i}} read(p) = u_{i+1} \text{ for all } 0 \leq i \leq s-1\}. \end{aligned}$$

Let Π be a genPCol automaton, and let $f : (V - \{e\})^* \to 2^{\Sigma^*}$ be a mapping, such that $f(u) = \{\varepsilon\}$ if and only if u is the empty multiset.

The *language accepted by* Π with respect to f is defined as

$$L(\Pi, f) = \{f(u_1)f(u_2)\dots f(u_s) \in \Sigma^* \mid \; u_1u_2\dots u_s \in A(\Pi)\}.$$

We define the following language classes.

- $\mathcal{L}(\text{genPCol}, \mathcal{F}, \text{com-tape}(k))$ is the class of languages accepted by generalized PCol automata with capacity k and with mappings from the class $\mathcal{F}$ where all the communication rules are tape rules,
- $\mathcal{L}(\text{genPCol}, \mathcal{F}, \text{all-tape}(k))$ is the class of languages accepted by generalized PCol automata with capacity k and with mappings from the class $\mathcal{F}$ where all the programs must have at least one tape rule,
- $\mathcal{L}(\text{genPCol}, \mathcal{F}, *(k))$ is the class of languages accepted by generalized PCol automata with capacity k and with mappings from the class $\mathcal{F}$ where programs with any kinds of rules are allowed.

Let $f : V^* \to 2^{\Sigma^*}$, for some alphabets V and Σ, and let the mapping f_{perm} and the class of mappings TRANS be defined as follows:

- $f = f_{perm}$ if and only if $V = \Sigma$ and for all $v \in V^{(*)}$, we have $f(v) = \{a_1a_2\dots a_s \mid |v| = s, \text{ and } a_1a_2\dots a_s \text{ is a permutation of the elements of } v\}$;
- $f \in$ TRANS if and only if for any $v \in V^{(*)}$, we have $f(v) = \{w\}$ for some $w \in \Sigma^*$ which is obtained by applying a finite transducer to the string representation of the multiset v, (as w is unique, the transducer must be constructed in such a way that all string representations of the multiset v as input result in the same $w \in \Sigma^*$ as output, and moreover, as f should be nonerasing, the transducer produces a result with $w \neq \varepsilon$ for any nonempty input).

We denote the above defined language classes as $\mathcal{L}_X(\text{genPCol}, Y(k))$, where $X \in \{f_{perm}, \text{TRANS}\},\; Y \in \{\text{com-tape}, \text{all-tape}, *\}$.

3 New Results on Systems with Input Mappings from TRANS

For any class of mappings $\mathcal{F}$, we have (see [6])

1. $\mathcal{L}(\text{genPCol}, \mathcal{F}, \text{com-tape}(k)) \subseteq \mathcal{L}(\text{genPCol}, \mathcal{F}, *(k))$ and $\mathcal{L}(\text{genPCol}, \mathcal{F}, \text{all-tape}(k)) \subseteq \mathcal{L}(\text{genPCol}, \mathcal{F}, *(k)$ for any $k \geq 1$; and
2. $\mathcal{L}(\text{genPCol}, \mathcal{F}, X(k)) \subseteq \mathcal{L}(\text{genPCol}, \mathcal{F}, X(k+1))$ for any $k \geq 1$ and $X \in \{\text{com-tape}, \text{all-tape}, *\}$.

The computational capacity of genPCol automata with input mappimg f_{perm} was investigated in [5] and [6]. It was shown that $\mathcal{L}_{perm}(\text{genPCol}, *(1)) = \mathcal{L}(RE)$, thus, it is not surprising, but the same holds also for the class of mappings TRANS.

Proposition 1.

$$\mathcal{L}_{\text{TRANS}}(\text{genPCol}, *(1)) = \mathcal{L}(RE).$$

A similar result holds for all-tape systems with capacity at least two. From [6] we have that $\mathcal{L}_{perm}(\text{genPCol}, \text{all-tape}(k)) = \mathcal{L}(\text{RE})$ for $k \geq 2$, and we can show the same for systems with input mappings from TRANS.

Proposition 2.

$$\mathcal{L}_{\text{TRANS}}(\text{genPCol}, \text{all-tape}(k)) = \mathcal{L}(\text{RE}) \textit{ for } k \geq 2.$$

For systems with capacity one, it is not difficult to see that all regular langugaes can be characterized, but a more precise characterization of the corresponding langugae classes are still missing.

Proposition 3.

$$\text{REG} \subseteq \mathcal{L}_{\text{TRANS}}(\text{genPCol}, \text{X}(1)), \textit{ for } \text{X} \in \{\text{all-tape}, \text{com-tape}\}.$$

The characterization of langugaes of com-tape systems is an interesting research direction. Similarly to systems with input mapping f_{perm}, we have the following, where r-1LOGSPACE denotes the class of languages characterized by so-called *restricted one-way logarithmic space* bounded Turing machines, see [3] for more on this complexity class.

Proposition 4.

$$\mathcal{L}_{\text{TRANS}}(\text{genPCol}, \text{com-tape}(2)) \subseteq \text{r-1LOGSPACE}.$$

As the class of languages characterized by P automata is strictly included in r-1LOGSPACE, the above statement does not give any information on the relationship of the power of P automata and genPCol automata. We know, however (see [6]), that genPCol automata with f_{perm} and com-tape programs can characterize languages that cannot be accepted by P automata using the mapping f_{perm}.

As P automata with sequential rule application and input mappings from TRANS characterize exctly the language class r-1LOGSPACE, the relationship of this language class and genPCol automata with input mappings from TRANS seems to be an especially interesting research direction.

Further, the effect of using *checking rules*, as defined in [8] for P colonies, is also an interesting topic for further investigations, just as the investigation of systems with other classes of input mappings besides f_{perm}.

References

1. L. Ciencialová, E. Csuhaj-Varjú, A. Kelemenová, Gy. Vaszil, Variants of P colonies with very simple cell structure. *International Journal of Computers Communication and Control*, **4** (2009), 224–233.

2. L. Cienciala, L. Ciencialová, E. Csuhaj-Varjú, Gy. Vaszil, *PCol automata: Recognizing strings with P colonies.* In: M. A. Martínez del Amor, Gh. Păun, I. Pérez Hurtado, A. Riscos Nuñez (eds.), Eighth Brainstorming Week on Membrane Computing, Sevilla, February 1-5, 2010, Fénix Editora, 2010, 65–76.
3. E. Csuhaj-Varjú, M. Oswald, Gy. Vaszil, P automata. In [10], chapter 6, 144–167.
4. E. Csuhaj-Varjú, Gy. Vaszil, P automata or purely communicating accepting P systems. In: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (eds.), *Membrane Computing. International Worskhop, WMC-CdeA 2002, Curtea de Arges, Romania, August 19-23, 2002, Revised Papers.* LNCS 2597, Springer Berlin Heidelberg, 2003, 219–233.
5. K. Kántor and Gy. Vaszil Generalized P Colony Automata *Journal of Automata, Languages and Combinatorics* **19** (2014), 145–156.
6. K. Kántor and Gy. Vaszil On the Class of Languages Characterized by Generalized P Colony Automata *Accepted*.
7. J. Kelemen, A. Kelemenová, A grammar-theoretic treatment of multiagent systems. *Cybernetics and Systems*, **23** (1992), 621–633.
8. J. Kelemen, A. Kelemenová, Gh. Păun, Preview of P colonies: A biochemically inspired computing model. In: M. Bedau et al. (eds.), *Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX).* Boston Mass., 2004, 82–86.
9. A. Kelemenová, P Colonies. In [10], chapter 23.1, 584–593.
10. Gh. Păun, G. Rozenberg, A. Salomaa, editors, *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.