VARIANTS OF P COLONY AUTOMATA

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

Abstract
We give an overview of P colony automata presenting recent results and research directions of the area.

1. Introduction

P colonies are tissue-like membrane systems modeling a community of very simple computing agents (cells), living together and interacting in a shared environment, see [17, 18]. The name *colony* comes from the theory of grammar systems (see [7]), from one of the grammatical models studied in the field called a *colony of grammars*. Such a colony (see also [16]), is a collection of very simple generative grammars (generating finite languages each), but by behaving as a cooperating system, they are able to generate fairly complicated languages, thus, the computing power of the system as a whole increases considerably when compared to the power of the individual components.

P colonies represent this approach in the framework of membrane computing. The model is similar to tissue-like membrane systems, the environment and the computing agents are both described by multisets of objects which are processed by the colony member cells using rules which enable the transformation of the objects and the exchange of objects between the cells and the environment. The capabilities of the computing agents are very restricted, both from the points of view of information storage and information processing possibilities. First, only a limited number of objects are allowed to be present inside the cells simultaneously (this is called the capacity of the system), and second, the rules are very simple, they are either of the form $a \to b$ (for changing an object a into an object b inside the cell), or $a \leftrightarrow b$ (for exchanging an object a inside a cell with an object b in the environment). The rules are grouped into programs. If the capacity of the colony is k, then each program consists of k rules which (when the program is applied) are applied to the k objects simultaneously. A computation of the colony consists of a sequence of computational steps during which the colony member cells execute their programs in parallel, until the system reaches one of the final configurations (usually given as the set of halting configurations, that is, those situations when no programs can be applied by any of the cells).

P colonies have been extensively studied, it has been shown, for example, that although they are extremely simple, they are computationally complete computing devices even with very restricted size parameters and other syntactic or functioning restrictions. For these, and more topics, results, see [5, 6, 4, 3, 8, 9, 12, 13].

As P colonies work with multisets of objects, it is natural to look at the result of the computations as sets of numbers (sets of vectors) represented by the multiplicities of certain objects present in the final configurations. On the other hand, being able to describe sets of strings instead of numbers, is also of interest. P colony automata were introduced in [2] for this purpose: to be able to accept with P colonies strings and string languages (instead of multiset collections). The idea of P colony automata is to assume the presence of an input tape with an input string, and designate certain rules as tape rules. When such a tape rule is applied, the symbol which is processed by the rule should also be read from the input tape. The difficulty of this idea comes from the possibility of applying several programs, and thus, several tape rules simultaneously, which gives rise to possible conflicts between tape rules which would need to read different symbols from the same input. Nevertheless, several variants of P colony automata turned out to be computationally complete, as shown in [2] and later in [1].

Here we consider a different way of dealing with strings in P colonies. The model was introduced in [15] under the name of generalized P colony automata, and studied further in [14]. The idea is to define the reading of input symbols in a way that is more close to the nature of other kinds of membrane computing systems, especially antiport P systems and P automata in particular. P automata, introduced in [11] (see also [10]), are P systems using symport and antiport rules (see [19]), characterizing string languages in a different way than "ordinary" P colony automata. They do not have input tapes with predefined input strings. Instead of reading input tapes, they associate strings to their computations by keeping track of the communication with the environment. They are not forced to a certain behavior through the given input, but operate and communicate freely with the environment, where each object which can be requested for input by the communication rules of the P system is assumed to be available in an unbounded supply. The accepted strings (the strings which are said to be read by the system) are determined by the sequences of those multisets which enter the system from the environment during computations. Such a sequence of multisets is mapped to a string, a sequence of symbols which constitute the string accepted by a particular computation.

A similar idea is employed in generalized P colony automata, the idea of characterizing strings through the sequences of multisets processed during computations. The computations of the colony define accepted multiset sequences, which are turned into accepted strings by mapping the multiset sequences to symbol sequences (strings) over some previously given alphabet. They also have rules distinguished as tape rules, and the application of such a tape rule also implies the reading of the processed symbol from the input (as in "ordinary" P colony automata), but unlike in the original model, the automaton is allowed to read more than one such symbol in a single computational step. This way generalized P colony automata avoid the conflicts that would arise by the simultaneous use of tape rules processing (and therefore reading) different symbols, but they may read several symbols (a multiset of symbols) in one computational step. This means that during a computation consisting of a sequence of computational steps, a sequence of multisets is read from the input. This sequence of multisets then can be mapped

into a string (a sequence of symbols) in a similar way as in P automata.

In [15], some basic variants of the model were introduced and studied from the point of view of their computational power. In [14] we continued the investigations structuring our results around the capacity of the systems, and different types of restrictions imposed on the use of tape rules in the programs of the systems. We considered three possible ways of dealing with tape rules in the programs: (1) the unrestricted case, (2) the case when all programs must contain at least one tape rule (all-tape programs), and (3) the case when all communication rules are tape rules (com-tape programs).

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 an *alphabet*, the alphabet of the automaton, its elements are called *objects*;
- $e \in V$ is the *environmental object* of the automaton, the only object which is assumed to be available in an arbitrary, unbounded number of copies in the environment;
- $w_E \in (V - \{e\})^*$ 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 $a \overset{T}{\to} b$, or $a \overset{T}{\leftrightarrow} b$, called rewriting tape rules and communication tape rules, respectively; or
  - *nontape rules* of the form $a \to b$, 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.
- 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 symbols) 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 $(u_E, u_1, \ldots, u_n)$, where $u_E \in (V - \{e\})^*$ represents the multiset of objects different from e in the environment, and $u_i \in V^*$, $1 \leq i \leq n$, represent the contents of the i-th cell. The *initial configuration* is given by

$(w_E, w_1, \ldots, w_n)$, 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 $(v_E, v_1, \ldots, v_n)$, 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-determinically 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 this way an accepted sequence of multisets: the sequence of multisets entering the system during the steps of the computation.

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

$$\begin{aligned} A(\Pi) = \{u_1 u_2 \ldots u_s \mid\ & u_i \in (V - \{e\})^*,\ 1 \leq i \leq s, \text{ and there is a configuration} \\ & \text{sequence } c_0, \ldots, c_s, \text{ with } c_0 = (w_E, w_1, \ldots, w_n),\ c_s \in F, \text{ and} \\ & c_i \Longrightarrow c_{i+1} \text{ with } \bigcup_{p \in P_{c_i}} read(p) = u_{i+1} \text{ for all } 0 \leq i \leq s-1\}. \end{aligned}$$

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

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

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 the mapping f_{perm} and the class of mappings TRANS be defined as follows:

- $f_{perm} : V^* \to 2^{\Sigma^*}$ where $V = \Sigma$ and for all $v \in V^*$, we have $f(v) = \{a_1 a_2 \ldots a_s \mid |v| = s,$ and $a_1 a_2 \ldots a_s$ is a permutation of the elements of $v\}$;

- for some $f : V^* \to 2^{\Sigma^*}$, we say that $f \in$ TRANS 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 these language classes as $\mathcal{L}_X(\text{genPCol}, Y(k))$, where $X \in \{f_{perm}, \text{TRANS}\}$, $Y \in \{\text{com-tape}, \text{all-tape}, *\}$.

Now we present an example to demonstrate the above defined notions.

Example 2.1 Let $\Pi = (\{a, b, c\}, e, \emptyset, (ea, P), F)$ be a genPCol automaton where

$$P = \{\langle e \to a, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to b, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to b, b \overset{T}{\leftrightarrow} a\rangle, \langle e \to c, b \overset{T}{\leftrightarrow} a\rangle, \\ \langle a \to b, b \overset{T}{\leftrightarrow} a\rangle, \langle a \to c, b \overset{T}{\leftrightarrow} a\rangle\}$$

with all the communication rules being tape rules. Let also $F = \{(\varepsilon; v, ca) \mid a \notin v\}$ be the set of final configurations. A possible computation of this system is the following:

$$(\emptyset, ea) \Rightarrow (a, ea) \Rightarrow (aa, ea) \Rightarrow (aaa, eb) \Rightarrow (aab, ba) \Rightarrow (bba, ba) \Rightarrow (bbb, ac)$$

where the first three computational steps read the multiset containing an a, the last three steps read a multiset containing a b, thus the accepted multiset sequence of this computation is $(a)(a)(a)(b)(b)(b)$.

It is not difficult to see that similarly to the one above, the computations which end in a final configuration (a configuration which does not containg the object a in the environment) accept the set of multiset sequences $A(\Pi) = \{(a)^n(b)^n \mid n \geq 1\}$.

If we consider f_{perm} as the input mapping, we have $L(\Pi, f_{perm}) = \{a^n b^n \mid n \geq 1\}$. On the other hand, if we consider a mapping $f_1 \in$ TRANS with $f_1 : \{a, b\}^* \to \{c, d, e, f\}^*$ and $f_1(a) = \{cd\}$, $f_1(b) = \{ef\}$ (and f_1 undefined in all other cases), we get the language $L(\Pi, f_1) = \{(cd)^n(ef)^n \mid n \geq 1\}$.

3. Recent Results on Systems with Input Mappings from TRANS

For any class of mappings $\mathcal{F}$, we have (see [14])

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

The computational capacity of genPCol automata with input mappimg f_{perm} was investigated in [15] and [14]. 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 3.1

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

A similar result holds for all-tape systems with capacity at least two. From [14] 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 3.2

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

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 3.3

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

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 [10] for more on this complexity class.

Proposition 3.4

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

4. Conclusions

As the class of languages characterized by P automata is strictly included in r-1LOGSPACE, the above theorem does not give any information on the relationship of the power of P automata and genPCol automata. We know, however, that genPCol automata with f_{perm} and com-tape programs are more powerful than P automata using the mapping f_{perm}.

As P automata with sequential rule application and input mappings from TRANS characterize exactly 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 [17] 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}.

Acknowledgements

This research was supported in part by grant no. MAT120558 of the National Research, Development and Innovation Office, Hungary.

References

[1] L. CIENCIALA, L. CIENCIALOVÁ, P Colonies and Their Extensions. In: J. KELEMEN, A. KELEMENOVÁ (eds.), *Computation, Cooperation, and Life – Essays Dedicated to Gheorghe Paun on the Occasion of His 60th Birthday*. Lecture Notes in Computer Science 6610, Springer, 2011, 158–169.

[2] L. CIENCIALA, L. CIENCIALOVÁ, E. CSUHAJ-VARJÚ, G. VASZIL, PCol automata: Recognizing strings with P colonies. In: M. A. MARTÍNEZ DEL AMOR, G. 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] L. CIENCIALA, L. CIENCIALOVÁ, A. KELEMENOVÁ, On the Number of Agents in P Colonies. In: G. ELEFTHERAKIS, P. KEFALAS, G. PAUN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25–28, 2007 Revised Selected and Invited Papers*. Lecture Notes in Computer Science 4860, Springer, 2007, 193–208.

[4] L. CIENCIALA, L. CIENCIALOVÁ, A. KELEMENOVÁ, Homogeneous P Colonies. *Computing and Informatics* 27 (2008) 3+, 481–496.

[5] L. CIENCIALOVÁ, L. CIENCIALA, Variation on the theme: P colonies. In: D. KOLĂR, A. MEDUNA (eds.), *Proc. 1st Intern. Workshop on Formal Models*. Ostrava, 2006, 27–34.

[6] L. CIENCIALOVÁ, E. CSUHAJ-VARJÚ, A. KELEMENOVÁ, G. VASZIL, Variants of P colonies with very simple cell structure. *International Journal of Computers, Communication and Control* 4 (2009) 3, 224–233.

[7] E. CSUHAJ-VARJÚ, J. DASSOW, J. KELEMEN, G. PĂUN, *Grammar Systems – A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach, London, 1994.

[8] E. CSUHAJ-VARJÚ, J. KELEMEN, A. KELEMENOVÁ, Computing with Cells in Environment: P Colonies. *Multiple-Valued Logic and Soft Computing* 12 (2006) 3–4, 201–215.

[9] E. CSUHAJ-VARJÚ, M. MARGENSTERN, G. VASZIL, P Colonies with a Bounded Number of Cells and Programs. In: H. J. HOOGEBOOM, G. PAUN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing, 7th International Workshop, WMC 2006, Leiden, The Netherlands, July 17–21, 2006, Revised, Selected, and Invited Papers*. Lecture Notes in Computer Science 4361, Springer, 2006, 352–366.

[10] E. CSUHAJ-VARJÚ, M. OSWALD, G. VASZIL, P automata. In: G. PAUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., 2010.

[11] E. CSUHAJ-VARJÚ, G. VASZIL, P Automata or Purely Communicating Accepting P Systems. In: G. PAUN, G. ROZENBERG, A. SALOMAA, C. ZANDRON (eds.), *Membrane Computing, International Workshop, WMC-CdeA 2002, Curtea de Arges, Romania, August 19–23, 2002, Revised Papers*. Lecture Notes in Computer Science 2597, Springer, 2002, 219–233.

[12] R. FREUND, M. OSWALD, P Colonies Working in the Maximally Parallel and in the Sequential Mode. In: D. ZAHARIE, D. PETCU, V. NEGRU, T. JEBELEAN, G. CIOBANU, A. CICORTAS, A. ABRAHAM, M. PAPRZYCKI (eds.), *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), 25–29 September 2005, Timisoara, Romania*. IEEE Computer Society, 2005, 419–426.

[13] R. FREUND, M. OSWALD, P colonies and prescribed teams. *Int. J. Comput. Math.* 83 (2006) 7, 569–592.

[14] K. KÁNTOR, G. VASZIL, On the Class of Languages Characterized by Generalized P Colony Automata. *Theoretical Computer Science* to appear.

[15] K. KÁNTOR, G. VASZIL, Generalized P Colony Automata. *Journal of Automata, Languages and Combinatorics* 19 (2014) 1–4, 145–156.

[16] J. KELEMEN, A. KELEMENOVÁ, A grammar-theoretic treatment of multiagent systems. *Cybernetics and Systems* 23 (1992), 621–633.

[17] J. KELEMEN, A. KELEMENOVA, G. PAUN, Preview of P colonies: A biochemically inspired computing model. In: *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE IX, Boston, Mass*. 2004, 82–86.

[18] A. KELEMENOVÁ, P colonies. In: G. PAUN, G. ROZENBERG, A. SALOMAA (eds.), *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., 2010, 584–593.

[19] A. PAUN, G. PĂUN, The Power of Communication: P Systems with Symport/Antiport. *New Generation Comput.* 20 (2002) 3, 295–306.