PARSING LANGUAGES OF P COLONY AUTOMATA

Erzsébet Csuhaj-Varjú[(A)] Kristóf Kántor[(B)]
György Vaszil[(B)]

[(A)]Department of Algorithms and Their Applications
Faculty of Informatics, ELTE Eötvös Loránd University,
Pázmány Péter sétány 1/c, 1117 Budapest, Hungary
csuhaj@inf.elte.hu

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

Abstract
In this paper a subclass of generalized P colony automata is defined that satisfies a property which resembles the LL(k) property of context-free grammars The possibility of parsing the characterized languages using a k symbol lookahead, as in the LL(k) parsing method for context-free languages, is examined.

1. Introduction

The computational model called P colony is similar to tissue-like membrane systems. In P colonies, multisets of objects are used to describe the contents of cells and the environment. These multisets are processed by the cells in the corresponding colony using rules which enable the evolution of the objects present in the cells or the exchange of objects between the environment and the cells. These cells or computing agents have a very restricted functionality: they can store a limited amount of objects at a given time (the capacity of the cell) and thus they can process a limited amount of information. For more information on P colonies, consult summaries [12, 2].

P colony automata were introduced in [3]. They are called automata, since these variants of P colonies accept string languages by assuming an initial input tape with an input string in the environment. The available types of rules are extended by so-called tape rules. These types of rules in addition to processing the objects as their non-tape counterparts, also read the processed objects from the input tape.

Generalized P colony automata were introduced in [9] to overcome the difficulty that different

tape rules can read different symbols in the same computational step. The main idea of this computational model was to get the process of input reading closer to other kinds of membrane systems, in particular to antiport P systems and P automata. The latter, introduced in [6] (see also [5]) are P systems using symport and antiport rules (see [13]), describing string languages. Generalized P colony automata were studied further in [11, 10].

A computation in this model defines accepted multiset sequences that are transformed into accepted symbol sequences/ strings. Generalized P colony automata have no input string, but there are tape rules and non-tape rules equally for evolution and communication rules. In a single computational step, this system is able to read more than one symbol, thus reading a multiset. This way generalized P colony automata are able to avoid the conflicts present in P colony automata, where simultaneous usage of tape rules in a single computational step can arise problems. After getting the result of a computation, that is, the accepted sequence of multisets, the sequence is mapped to a string in a similar way as shown in P automata.

In [9], some basic variants of the model were introduced and studied from the point of view of their computational power. In [11, 10] the investigations were continued by structuring the previous results around the capacity of the systems, and different types of restrictions imposed on the use of tape rules in the programs.

Since P colony automata variants accept languages, different types of descriptions of their language classes are of interest. One possible research direction is to investigate their parsing properties in terms of programs and rules of the (generalized) P colony automata. In this paper, we study the possibility of deterministically parsing the languages characterized by these devices. We define the so-called LL(k) condition for these types of automata, which enables deterministic parsing with a k symbol lookahead as in the case of context-free LL(k) languages. As an initial result, we show that using generalized P colony automata we can deterministically parse context-free languages that are not LL(k) in the "original" sense.

An extended version of this short paper has been submitted for publication, see [4].

2. Preliminaries and Definitions

Let V be a finite alphabet, let the set of all words over V be denoted by V^*, and let ε be the empty word. We denote the cardinality of a finite set S by $|S|$, and the number of occurrences of a symbol $a \in V$ in w by $|w|_a$.

A multiset over a set V is a mapping $M : V \to \mathbb{N}$ where $\mathbb{N}$ denotes the set of non-negative integers. This mapping assigns to each object $a \in V$ its multiplicity $M(a)$ in M. The set $supp(M) = \{a \mid M(a) \geq 1\}$ is the support of M. If V is a finite set, then M is called a finite multiset. A multiset M is empty if its support is empty, $supp(M) = \emptyset$. The set of finite multisets over the alphabet V is denoted by $\mathcal{M}(V)$. A finite multiset M over V will also be represented by a string w over the alphabet V with $|w|_a = M(a)$, $a \in V$, the empty multiset will be denoted by $\emptyset$.

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 a multiset from $\mathcal{M}(V - \{e\})$, 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 (the representation of) 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 $a \xrightarrow{T} b$, or $a \xleftrightarrow{T} b$, 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.
- 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 $(u_E, u_1, \ldots, u_n)$, where $u_E \in \mathcal{M}(V - \{e\})$ is the multiset of objects different from e in the environment, and $u_i \in \mathcal{M}(V),\ 1 \leq i \leq n$, are 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 $v_E \in \mathcal{M}(V - \{e\})$ denotes a multiset of objects different from e being in the environment, and $v_i \in \mathcal{M}(V),\ 1 \leq i \leq n$, is the contents of the i-th cell.

Let $c = (u_E, u_1, \ldots, u_n)$ be a configuration of a genPCol automaton Π, and let $U_E = u_E \cup \{e, e, \ldots\}$, thus, the multiset of objects found in the environment (together with the infinite number of copies of e, denoted as $\{e, e, \ldots\}$, which are always present). The *sequence of programs*

$$(p_1, \ldots, p_n) \in (P_1 \cup \{\#\}) \times \ldots \times (P_n \cup \{\#\})$$

is *applicable in configuration* c, if the following conditions hold: (1) The selected programs are applicable in the cells, (2) the symbols to be brought inside the cells by the programs are present in the environment, (3) the set of selected programs is maximal.

Let us denote the applicable sequences of programs in the configuration $c = (u_E, u_1, \ldots, u_n)$ by App_c, that is,

$$App_c = \{P_c = (p_1, \ldots, p_n) \in (P_1 \cup \{\#\}) \times \ldots \times (P_n \cup \{\#\}) \mid \text{ where } P_c \text{ is a sequence of applicable programs in the configuration } c\}.$$

A configuration c is called *a halting configuration* if the set of applicable sequences of programs is the singleton set $App_c = \{(p_1, \ldots, p_n) \mid p_i = \# \text{ for all } 1 \leq i \leq n\}$.

Let $c = (u_E, u_1, \ldots, u_n)$ be a configuration of the genPCol automaton. By applying a sequence of applicable programs $P_c \in App_c$, the configuration c is *changed* to a configuration $c' = (u'_E, u'_1, \ldots, u'_n)$, denoted by $c \overset{P_c}{\Longrightarrow} c'$, if the following properties hold. (For a program p, we denote by $create(p)$, $import(p)$, and $export(p)$ the multisets of objects created by the program through rewriting, brought inside the cell from the environment, and sent out to the environment, respectively.)

- If $(p_1, \ldots, p_n) = P_c \in App_c$ and $p_i \in P_i$, then $u'_i = create(p_i) \cup import(p_i)$, otherwise, if $p_i = \#$, then $u'_i = u_i$, $1 \leq i \leq n$. Moreover,
- $U'_E = U_E - \bigcup_{p_i \neq \#, 1 \leq i \leq n} import(p_i) \cup \bigcup_{p_i \neq \#, 1 \leq i \leq n} export(p_i)$ (where U'_E again denotes $u'_E \cup \{e, e, \ldots\}$ with an infinite number of copies of e).

Thus, in genPCol automata, we apply the programs in the maximally parallel way, that is, in each computational step, every component cell nondeterministically applies one of its applicable programs. Then we collect all the symbols that the tape rules "read": this is the multiset read by the system in the given computational step.

For any P_c sequence of applicable programs in a configuration c, let us denote the multiset of objects read by the tape rules of the programs of P_c by $read(P_c)$. Then we can also define the set of multisets which can be read in any configuration of the genPCol automaton Π as

$$in_c(\Pi) = \{read(P_c) \mid P_c \in App_c\}.$$

Remark 2.1 *Although the set of configurations of a genPCol automaton Π can be infinite (because the multiset corresponding to the contents of the environment is not necessarily finite), the set $in_c(\Pi)$ is always finite.*

A successful computation defines this way an accepted sequence of multisets: $u_1 u_2 \ldots u_s$, $u_i \in in_{c_{i-1}}(\Pi)$, for $1 \leq i \leq s$, that is, 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

$$A(\Pi) = \{u_1 u_2 \ldots u_s \mid u_i \in in_{c_{i-1}}(\Pi),\ 1 \leq i \leq s, \text{ and there is a configuration sequence } c_0, \ldots, c_s, \text{ with } c_0 = (w_E, w_1, \ldots, w_n),\ c_s \in F,\ c_s \text{ halting, and } c_i \overset{P_{c_i}}{\Longrightarrow} c_{i+1} \text{ with } u_{i+1} = read(P_{c_i}) \text{ for all } 0 \leq i \leq s-1\}.$$

Let Π be a genPCol automaton, and let $f : \mathcal{M}(V) \to 2^{\Sigma^*}$ 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

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

Let V and Σ be two alphabets, and let $\mathcal{M}_{FIN}(V) \subseteq \mathcal{M}(V)$ denote the set of finite subsets of the set of finite multisets over an alphabet V. Consider a mapping $f : D \to 2^{\Sigma^*}$ for some $D \subseteq \mathcal{M}_{FIN}(V)$. We say that $f \in \mathcal{F}_{\text{TRANS}}$, if for any $v \in D$, we have $|f(v)| = 1$, and we can obtain $f(v) = \{w\}$, $w \in \Sigma^*$ by applying a deterministic finite transducer to any 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).

Besides the above defined class of mappings, we also use the so-called permutation mapping. Let $f_{perm} : \mathcal{M}(V) \to 2^{\Sigma^*}$ where $V = \Sigma$ be defined as follows. For all $v \in \mathcal{M}(V)$, we have

$$f_{perm}(v) = \{a_{\sigma(1)}a_{\sigma(2)}\dots a_{\sigma(s)} \mid v = a_1a_2\dots a_s \text{ for some permutation } \sigma\}.$$

3. P Colony Automata and the LL(k) Condition

Let $U \subset \Sigma^*$ be a finite set of strings over some alphabet Σ. Let us denote for some $k \geq 1$, the set of length k prefixes of the elements of U by $\text{FIRST}_k(U)$, that is, let

$$\text{FIRST}_k(U) = \{pref_k(u) \in \Sigma^* \mid u \in U\}$$

where $pref_k(u)$ denotes the string of the first k symbols of u if $|u| \geq k$, or $pref_k(u) = u$ otherwise.

Definition 3.1 Let $\Pi = (V, e, w_E, (w_1, P_1), \dots, (w_n, P_n), F)$ be a genPCol automaton, let $f : \mathcal{M}(V) \to 2^{\Sigma^*}$ be a mapping as above, and let $c_0, c_1, \dots, c_s$ be a sequence of configurations with $c_i \Longrightarrow c_{i+1}$ for all $0 \leq i \leq s-1$.

We say that the P colony Π is LL(k) for some $k \geq 1$ with respect to the mapping f, if for any two distinct sets of programs applicable in configuration c_s, $P_{c_s}, P'_{c_s} \in App_{c_s}$ with $P_{c_s} \neq P'_{c_s}$, the next k symbols of the input string that is being read determines which of the two sequences are to be applied in the next computational step, that is, the following holds.

Consider two computations

$$c_s \overset{P_{c_s}}{\Longrightarrow} c_{s+1} \overset{P_{c_{s+1}}}{\Longrightarrow} \dots \overset{P_{c_{s+m}}}{\Longrightarrow} c_{s+m+1}, \text{ and } c_s \overset{P'_{c_s}}{\Longrightarrow} c'_{s+1} \overset{P_{c'_{s+1}}}{\Longrightarrow} \dots \overset{P_{c'_{s+m'}}}{\Longrightarrow} c'_{s+m'+1}$$

where $u_{c_s} = read(P_{c_s})$ and $u_{c_{s+i}} = read(P_{c_{s+i}})$ for $1 \leq i \leq m$, and similarly $u'_{c_s} = read(P'_{c_s})$ and $u_{c'_{s+i}} = read(P_{c'_{s+i}})$ for $1 \leq i \leq m'$, thus, the two sequences of input multisets are

$$u_{c_s}u_{c_{s+1}}\dots u_{c_{s+m}} \text{ and } u'_{c_s}u_{c'_{s+1}}\dots u_{c'_{s+m'}}.$$

Assume that these sequences are long enough to "consume" the next k symbols of the input string, that is, for w and w' with

$$w \in f(u_{c_s})f(u_{c_{s+1}}) \dots f(u_{c_{s+m}}) \text{ and } w' \in f(u'_{c_s})f(u_{c'_{s+1}}) \dots f(u_{c'_{s+m'}}),$$

either $|w| \geq k$ and $|w'| \geq k$, or if $|w| < k$ (or $|w'| < k$), then c_{s+m+1} (or $c'_{s+m'+1}$) is a halting configuration.

The P colony Π is LL(k), if for any two computations as above, $\text{FIRST}_k(w) \cap \text{FIRST}_k(w') = \emptyset$.

Let us illustrate the above definition with an example.

Example 3.2 *Let* $\Pi = (\{a, b, c, d, f, g, e\}, e, \emptyset, (ea, P_1), F)$ *where*

$$\begin{aligned} P_1 = \{&\langle e \to b, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to e, b \overset{T}{\leftrightarrow} a\rangle, \langle e \to c, a \overset{T}{\leftrightarrow} e\rangle, \langle e \to f, a \overset{T}{\leftrightarrow} e\rangle, \\ &\langle e \to d, c \overset{T}{\leftrightarrow} b\rangle, \langle b \to c, d \overset{T}{\leftrightarrow} e\rangle, \langle e \to g, f \overset{T}{\leftrightarrow} b\rangle, \langle b \to f, g \overset{T}{\leftrightarrow} e\rangle\} \text{ and} \\ F = \{&(v, ce), (v, fe) \mid v \in V^*, b \notin v\}. \end{aligned}$$

The language characterized by Π *is*

$$L(\Pi, f_{perm}) = \{a\} \cup \{(ab)^n a(cd)^n \mid n \geq 1\} \cup \{(ab)^n a(fg)^n \mid n \geq 1\}.$$

To see this, consider the possible computations of Π. *The initial configuration is* $(\emptyset, ea)$ *and there are three possible configurations that can be reached. Two of these are non-accepting states, but the derivations cannot be continued, so let us consider the third one (we denote by* $\Rightarrow_u$ *a configuration change during which the multiset of symbols* u *was read by the automaton).*

$$(a, be) \Rightarrow_b (b, ea) \Rightarrow_a (ba, be) \Rightarrow_b (bb, ea) \Rightarrow_a \dots \Rightarrow_b (b^i, ea).$$

At this point, the computation can follow two different paths again, either

$$(b^i, ae) \Rightarrow_a (b^i a, ec) \Rightarrow_c (b^{i-1}ac, db) \Rightarrow_d (b^{i-1}acd, ce) \Rightarrow_c \dots \Rightarrow_d (ac^i d^i, ce),$$

or

$$(b^i, ae) \Rightarrow_a (b^i a, ef) \Rightarrow_f (b^{i-1}af, gb) \Rightarrow_g (b^{i-1}afg, fe) \Rightarrow_f \dots \Rightarrow_g (af^i g^i, fe).$$

In the first phase of the computation, the system produces copies of b *and sends them to the environment, then in the second phase these copies of* b *are exchanged to copies of* cd *or copies of* fg. *The system can reach an accepting state when all the copies of* b *are used, that is, when an equal number of copies of* ab *and either of* cd *or of* fg *were produced.*

Note that the system satisfies the LL(1) property, the symbol that has to be read, in order to accept a desired input word, determines the set of programs that has to be used in the next computational step.

Let us denote the class of context-free LL(k) languages by $\mathcal{L}$(CF,LL(k)) (see for example the monograph [1] for more details) and the languages characterized by genPCol automata satisfying the above defined condition with input mapping of type f_{perm} or $f \in TRANS$, as $\mathcal{L}_X$(genPCol,LL(k)), $X \in \{perm, TRANS\}$.

The following statement can be presented.

Theorem 3.3 *There are context-free languages in $\mathcal{L}_X$(genPCol,LL(1)), $X \in \{perm, TRANS\}$, which are not in $\mathcal{L}$(CF,LL(k)) for any $k \geq 1$.*

Proof. The language $L(\Pi, f_{perm}) \in \mathcal{L}_{perm}$(genPCol,LL(1)) from Example 3.2 is not in $\mathcal{L}$(CF,LL(k)) for any $k \geq 1$. If we consider the mapping $f_1 \in TRANS$, $f_1 : \{a, b, c, d, f, g\} \to \{a, b, c, d, f, g\}$ with $f_1(x) = x$ for all $x \in \{a, b, c, d, f, g\}$, then $L(\Pi, f_1) = L(\Pi, f_{perm})$, thus, $\mathcal{L}_{TRANS}$(genPCol,LL(1)) also contains the non-LL(k) context-free language. □

4. Conclusions

P systems and their variants are able to describe powerful language classes, thus their applicability in the theory of parsing or analyzing syntactic structures are of particular interest, see, for example [7, 8]. In [7], so-called active P automata (P automata with dynamically changing membrane structure) were used for parsing, utilizing the dynamically changing membrane structure of the P automaton for analyzing the string. In this paper we studied the possibility of deterministically parsing languages characterized by P colony automata. We provided the definition of an LL(k)-like property for (generalized) P colony automata, and showed that languages which are not LL(k) in the "original" context-free sense for any $k \geq 1$ can be characterized by LL(1) P colony automata with different types of input mappings. The properties of these language classes for different values of k and different types of input mappings are open to further investigations.

Acknowledgments. The work of E. Csuhaj-Varjú was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558. The work of K. Kántor and Gy. Vaszil was supported in part by the National Research, Development and Innovation Office of Hungary, NKFIH, grant no. K 120558 and also by the construction EFOP-3.6.3-VEKOP-16-2017-00002, a project financed by the European Union, co-financed by the European Social Fund.

References

[1] A. V. AHO, J. D. ULMANN, *The Theory of Parsing, Translation, and Compiling*. 1, Prentice-Hall, Englewood Cliffs, N.J., 1973.

[2] L. CIENCIALA, L. CIENCIALOVÁ, E. CSUHAJ-VARJÚ, P. SOSÍK, P colonies. *Bulletin of the International Membrane Computing Society* 1 (2016) 2, 119–156.

[3] 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.

[4] E. CSUHAJ-VARJÚ, K. KÁNTOR, G. VASZIL, Deterministic Parsing with P Colony Automata. Submitted.

[5] 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.

[6] 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.

[7] G. B. ENGUIX, R. GRAMATOVICI, Parsing with Active P Automata. In: C. MARTÍN-VIDE, G. MAURI, G. PAUN, G. ROZENBERG, A. SALOMAA (eds.), *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Revised Papers*. Lecture Notes in Computer Science 2933, Springer, 2003, 31–42.

[8] G. B. ENGUIX, B. NAGY, Modeling Syntactic Complexity with P Systems: A Preview. In: O. H. IBARRA, L. KARI, S. KOPECKI (eds.), *Unconventional Computation and Natural Computation - 13th International Conference, UCNC 2014, London, ON, Canada, July 14-18, 2014, Proceedings*. Lecture Notes in Computer Science 8553, Springer, 2014, 54–66.

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

[10] K. KÁNTOR, G. VASZIL, Generalized P Colony Automata and Their Relation to P Automata. In: M. GHEORGHE, G. ROZENBERG, A. SALOMAA, C. ZANDRON (eds.), *Membrane Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers*. Lecture Notes in Computer Science 10725, Springer, 2017, 167–182.

[11] K. KÁNTOR, G. VASZIL, On the classes of languages characterized by generalized P colony automata. *Theor. Comput. Sci.* 724 (2018), 35–44.

[12] 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.

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