On PCol Automata working in the *t*-mode[⋆]

Luděk Cienciala and Lucie Ciencialová

Institute of Computer Science, Silesian University in Opava, Czech Republic
{`ludek.cienciala, lucie.ciencialova`}`@fpf.slu.cz`

1 Introduction and definitions

PCol automata are constructed as extension of P colonies following the notion of automata-like abstract models. P colonies and PCol automata are computational models belonging to the family of P systems. More can reader find in [12, 13].

The model of P colony is formed from simple cells living together in a shared environment. The cell, sometimes called agent, contains given number of objects. To execute the programs from its set of programs it uses internal objects and objects from environment(external objects). The programs of cells consist of rules which are either of the form $a \to b$, specifying that an internal object a is transformed into an internal object b, or of the form $c \leftrightarrow d$, specifying the fact that an internal object c is sent out of the cell, to the environment, in exchange of the object d, which was present in the environment. For more about P colonies we refer to [10, 11]. P colonies have been extensively examined during the last years: it has been shown that these extremely simple constructs are computationally complete computing devices even with very restricted size parameters and with other syntactic or functioning restrictions [3, 1, 2, 5, 6, 8].

In reference to finite automaton P colony was extended by a input tape and it changes the generating device into the accepting one ([4]). The cells of this kind of P colony are working according to actually read symbol from the input tape. To do this they have programs containing rule which can "read" the input tape. The cells execute programs in such a way to obtain the same kind of object as they read from input.

Definition 1. A *PCol 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 PCol automaton, its elements are called *objects*; $e \in V$ is the *environmental object* of the automaton; $w_E \in (V - \{e\})^*$ is a string representing the multiset of objects different from e which is found in the environment initially; $(w_i, P_i), 1 \leq i \leq n$, is the i-th *cell*; and F is a set of *accepting configurations* of the PCol automaton. For each cell, (w_i, P_i), $1 \leq i \leq n$, 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*, where every program is formed from k rules of the following types: (1) *tape rules* of the form $a \xrightarrow{T} b$, or

⋆ Research supported in part by the Czech Science Foundation, "GAČR", project 201/09/P075.

$a \overset{T}{\leftrightarrow} b$, called rewriting tape rules and communication tape rules, respectively; or (2) *nontape rules* of the form $a \to b$, or $c \leftrightarrow d$, called rewriting (nontape) rules and communication (nontape) rules, respectively. For each $i,\ 1 \leq i \leq n$, the set of *tape programs* is denoted by P_i^T, they are formed from one tape rule and $k-1$ nontape rules, the set of *nontape programs* which contain only nontape rules, is denoted by P_i^N, thus, $P_i = P_i^T \cup P_i^N,\ P_i^T \cap P_i^N = \emptyset$.

A *configuration* of PCol automaton is an $(n+2)$-tuple $(u; u_E, u_1, \ldots, u_n)$, where $u \in V^*$ is the unprocessed (unread) part of the input string, $u_E \in (V - \{e\})^*$ 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 i-th cell. The *initial configuration* is given by $(w; w_E, w_1, \ldots, w_n)$, the input word to be processed by the system and 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 $(\varepsilon; v_E, v_1, \ldots, v_n)$.

The *computation* of a PCol automaton starts in the initial configuration, and the configurations are changed by the cells with the application of some of their programs. During the computation, the PCol automaton processes an input word. The leftmost symbol of the non-read part of the input word is read during a computation if at least one cell applies a tape program which introduces the same symbol inside the cell as the symbol to be read either by rewriting or by communication.

PCol automaton can work in many modes according to using of tape rules. One of these modes is t-mode. In each step cells can execute only programs with tape rule. If cell has no such program to follow actually read symbol from the input tape, it cannot execute any of its programs. For detailed definition of computation and more about working modes we refer to [4].

2 The capacity and the number of cells

In this section we want to examine how capacity and number of cells affect the computation and language accepted by PCol automata. For the next comparison we need to refer to the definition of counter automaton [9]

Definition 2. A k-counter automaton $M = (Q, \Sigma, \delta, q_0, F)$ consists of a finite set Q of states, a designated initial state q_0, a set F of final or accepting states with $F \subseteq Q$, a finite set Σ of input symbols, and a transition function $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \{0, 1, -1\} \to P(Q \times \{0, 1, -1\}^k)$. A k-counter automaton is partially blind if for all $q \in Q$ and $a \in \Sigma \cup \{\varepsilon\}$ we have that $\delta(q, a, u_1, \ldots, u_k) = \emptyset$ whenever any u_i is -1, and $\delta(q, a, u_1, \delta, u_k) = \delta(q, a, v_1, \delta, v_k)$ for all $u_i, v_i \in \{0, 1\}$.

A configuration of M is an element of $Q \times \Sigma^* \times \mathbb{Z}^k$. If $(q', v_1, \ldots, v_k)$ is in $\delta(q, a, u_1, \ldots, u_k)$ and $(q, aw, y_1, \ldots, y_k)$ is configuration with $sgn(u_i) = sgn(y_i)$ for $1 \leq i \leq k$, (where for an integer x $sgn(x) = 1$, 0, or -1 if $x > 0$, $x = 0$, or $x < 0$, respectively), then we write $(q, aw, y_1, \ldots, y_k) \vdash (q', w, y_1{+}v_1, \ldots, y_k{+}v_k)$. If $a = \varepsilon$ the transition is called an ε-transition. The language accepted by M is $L(M) = \{w \in \Sigma^* | (q_0, w, 0, \ldots, 0) \vdash^* (q, \varepsilon, 0, \ldots, 0)$ for any $q \in F\}$.

Theorem 1. *For any partially blind* 1*-counter automaton M without ε-transitions, there exists PCol automaton Π with capacity $c = 4$ such that $L(M) = L(\Pi)$.*

Proof. (Sketch) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a partially blind 1-counter automaton without ε-transitions.

We construct PCol automaton $\Pi = (V, e, w_E, (w_1, P_1), F')$ such that $L(M) = L(\Pi)$, where $w_e = \varepsilon$ - the initial content of environment w_e is formed only from copies of object e; $w_1 = q_0eec$ - the initial content of the cell at the beginning of computation; $F' = \{(\varepsilon,\ \varepsilon,\ q_f\alpha\beta c) \mid q_f \in F, \alpha \in \Sigma, \beta \in \{e, c\}\}$; for all $q_r, q_s \in Q$, $a \in \Sigma$ and $x \in \{0, 1, -1\}$, if $(q_s, x) \in \delta(q_r, a)$, we add to P_1 following programs:

(1) $\left\langle q_r \xrightarrow{T} a; \alpha \rightarrow q_s; \beta \leftrightarrow c; c \rightarrow c \right\rangle$ if $x = -1$;

(2) $\left\langle q_r \xrightarrow{T} a; \alpha \rightarrow q_s; \beta \rightarrow c; c \leftrightarrow e \right\rangle$ if $x = 1$ and

(3) $\left\langle q_r \xrightarrow{T} a; \alpha \rightarrow q_s; \beta \rightarrow e; c \rightarrow c \right\rangle$ if $x = 0$, where $\alpha \in \Sigma \cup \{e\},\ \beta \in \{e, c\}$.

The programs in the cell are executed in the same sequence and have the same effect to configuration of the cell and to the number of objects c placed in the environment as transitions have to the state of counter automaton M and to the number stored in the counter.

Corollary 1. *To every partially blind n-counter automaton M without ε-transitions there exists PCol automaton Π with capacity $2n+2$ and one cell working in t-mode such that $L(M) = L(\Pi)$. To every partially blind deterministic n-counter automaton M without ε-transitions there exists PCol automaton Π with capacity* 4 *and n cells working in t-mode such that $L(M) = L(\Pi)$.*

Theorem 2. *The family of languages accepted by PCol automata with one cell working in t-mode are subset of context-sensitive languages.*

Proof. (Idea) In the first phase Turing machine replaces symbols in the tape by new symbols formed from two parts. The first part corresponds to the original symbol in the tape, the second part describes objects in the environment. The content of the cell will be coded in the state of Turing machine. The environmental symbols we don't take into account. Turing machine simulates executing the programs as follows: it nondeterministically chooses one of possibly applicable programs (with information about content of cell and actual input symbol); if rules in the program are rewriting Turing machine changes state and continues with choosing another program; if one or more rules are communication, machine looks for external objects to exchange them with internal objects; if Turing machine successfully simulates executing of the program it marks the current input symbol as read; if simulation cannot be finished, machine chooses program again. When the cell is in the final configuration, machine controls unread symbol and objects in the environment. It stops the computation if the PCol automaton is in the final configuration or if there is some unread symbol continue computation.

3 Conclusion

This contribution is dedicated to PCol automata with one cell working in t-mode. We set some boundaries of family of languages accepted by this type of PCol automata - this family is subset of context-sensitive languages. The languages which can be accepted by partially blind one-counter automata without ε transitions can be accepted with PCol automata with one cell working in t-mode.

References

1. Cienciala, L., Ciencialová, L., Kelemenová, A.: Homogeneous P colonies. Computing and Informatics 27, 481-496 (2008)
2. Cienciala, L. Ciencialová, L., Kelemenová, A.: On the number of agents in P colonies. In: Eleftherakis G. et. al.(eds.) Membrane Computing. 8th International Workshop, WMC 2007. Thessaloniki, Greece, June 25-28, 2007., LNCS 4860, Springer-Verlag, Berlin-Heidelberg, 193-208 (2007).
3. Ciencialová, L., Csuhaj-Varjú, E., Kelemenová, A., Vaszil, Gy.: Variants of P colonies with very simple cell structure. International Journal of Computers, Communication and Control 4(3), 224-233 (2009)
4. Ciencialová, L. Cienciala, L., Csuhaj-Varjú, E., Vaszil, Gy.: PCol Automata: Recognizing Strings with P Colonies, Report of Eight Brainstorming week on membrane computing, Sevilla (2010)
5. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, Gh., Vaszil, Gy.: Computing with cells in environment: P colonies. Journal of Multi-Valued Logic and Soft Computing 12, 201-215 (2006)
6. Csuhaj-Varjú, E., Margenstern, M., Vaszil, Gy.: P colonies with a bounded number of cells and programs. In: Hoogeboom, H-J. et. al.(eds.) Membrane Computing. 7th International Worskhop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006. LNCS 4361, Springer-Verlag, Berlin-Heidelberg, pp. 352-366 (2007)
7. Csuhaj-Varjú, E., Oswald, M., Vaszil, Gy.: P automata. Chapter 6, In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) The Oxford Handbook of Membrane Computing, Oxford University Press, 144-167 (2010)
8. Freund, R., Oswald, M.: P colonies working in the maximally parallel and in the sequential mode. Pre-Proc. In: Ciobanu, G., Păun, Gh. (eds.) 1st Intern. Workshop on Theory and Application of P Systems, Timisoara, Romania, 49-56 (2005)
9. Greibach, S.: Remarks on Blind and Partially Blind One-way Multicounter Machines. Theoretical Computer Science 7, 311-324 (1978)
10. Kelemen, J., Kelemenová, A., Păun, Gh.: Preview of P colonies: A biochemically inspired computing model. In: Bedau, M. et. al.(eds.) Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX), Boston Mass, 82-86 (2004)
11. Kelemenová, A.: P Colonies. Chapter 23.1, In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) The Oxford Handbook of Membrane Computing, Oxford University Press, 584-593 (2010)
12. Păun, Gh.: Membrane Computing – An Introduction. Springer-Verlag, Berlin (2002)
13. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing, Oxford University Press, (2010)