数理解析研究所講究録
1166 巻 2000 年 178-190

Parallel Communicating Finite Automata Systems

Victor MITRANA[1]
University of Bucharest, Faculty of Mathematics
Str. Academiei 14, 70109, Bucharest, Romania
E-mail: mitrana@funinf.math.unibuc.ro

Abstract. A parallel communicating finite automata system is an accepting device based on the communication between more finite automata working in parallel. It consists of several automata working independently but communicating with each other by request. We survey several variants of parallel communicating finite automata systems with respect to their computational power. Other aspects like decidability and complexity matters are also briefly discussed. Some open problems and directions for future research are finally pointed out.

1 Introduction

In many areas of computer science (parallel computers, computer networks, DNA computing, artificial intelligence) many models based on cooperation and communication among agents have been considered. The formal language theory has been involved in most of these circumstances e.g. for modelling aspects whose essence can be captured at the level of abstract symbol systems, see, e.g., [5], [8], [21], etc. Thus, [4] introduced the concept of *system of grammars*, motivated by the so called "blackboard model" in problem solving theory [8]. More grammars working together, following a prescribed strategy is a grammar system. The same architecture is proposed in [1] with motivations coming from regulated rewriting area. Two essentially different architectures are known depending on the protocols of cooperation and communication among the components of the system, see, e.g, [5]. In the case of *cooperating distributed grammar systems* the cooperation is done by means of the sentential form; all components may rewrite, in turn, the sentential form accordingly to their own strategies. When a component is active, all the others are inactive. Quite different is the cooperation in *parallel communicating grammar systems*, where the components are working in parallel, and from time to time some components ask, by means of some query symbols,

[1]Research supported by the Dirección General de Enseñanza Superior e Investigación Cientifica, SB 97–00110508

for the work of other ones. The contacted components have to send their current work (sentential form) to those components which asked for it.

Systems of cooperating automata have also been considered as models for some computing systems, but the strategies of coordinate their work in order to perform some computation were very different than those considered in the grammar systems theory. Without the aim of completeness, we briefly mention some important models.

A *multiprocessor automaton* consists of several finite automata, called *processors* [2], which are coordinated by a central processing unit that decides which processor is to become active or "frozen" at a given step. Each processor works independently from the other ones according to its internal transition function which depends on the internal state of the processor and the current input symbol. The central processing unit inspects the current states of the processors (a frozen processor preserves its internal state and reading head position) and determines which processors will be active or frozen at the next step. Note that the states achieved by the processors depend exclusively on their current state and input symbol. The strategy of cooperation takes into consideration all internal states at a given step being limited to timing through which the central unit lets some processors proceed.

In another model each automaton is allowed to know the states of all automata. The transition function of one automaton depends on the input symbol currently read and the states of all automata determining a move of its reading head and a new state. An equivalent form of this system is the *multi-head automaton* which reduces all components to just reading heads controlled by a single processing unit with finitely many internal states.

The first approach of considering automata systems working under similar strategies to those defined for grammar systems can be found in [7], where the strategies considered for multistack pushdown automata are similar to those defined for cooperating distributed grammar systems.

Parallel communicating finite automata systems are somewhere in between these extremes. Their components are finite automata working independently but communicating states to each other by request. These systems, whose components communicate with each other under similar protocols to those considered for parallel communicating grammar systems [20], [5], have been introduced in [18]. Every component is entitled to request the state of any other component; the contacted component communicates its current state and remains in the same state (in the non-returning strategy) or enters again the initial state (in the returning strategy). In centralized systems only one component (the master of the system) is allowed to ask a state from the others. We want to stress that each step in an automata system is either a usual accepting step or a communication one; moreover, the communication steps have priority to the accepting ones. We also mention that whenever a component requests a state, the state must be communicated.

The investigation of the effect of these strategies of communication for systems of automata whose components are pushdown automata was continued in [6]. Thus, one distinguished two possible directions: communication by states or by stacks. Since one can easily observe that every two-stack pushdown automaton [12] can be simulated by a parallel communicating pushdown automata system whose components communicate states to each other, [6] focused its attention on the other way of communication, namely, communication by stacks.

The paper is organized as follows. The next section starts with the definitions of parallel communicating finite automata systems and their cooperation protocols, illustrated by a few examples. Then, we recall the main results known concerning the computational power of these devices, some decidability problems and a brief discussion related to mildly context-sensitive formalisms. Afterwards, we introduce a complexity measure for these systems called the degree of communication and investigate some computational aspects of this measure. Each section ends by briefly discussing some open problems and directions for future research.

2 Definitions and examples

We shall assume the reader familiar with the fundamental concepts of formal language theory and automata theory, particularly the notions of grammars and finite automata [22].

An alphabet is always a finite set of letters. The set of all words over an alphabet V is denoted by V^*. The empty word is written ε; moreover, $V^+ = V^* - \{\varepsilon\}$. For a finite set A we denote by $card(A)$ the cardinality of A. Sometimes, for a given alphabet V and a word $x = a_1a_2\ldots a_n$, $a_i \in V, 1 \leq i \leq n$, we write $\bar{V} = \{\bar{a} | a \in V\}$ and $\bar{x} = \bar{a}_1\bar{a}_2\ldots\bar{a}_n$.

A *parallel communicating finite automata system* of degree n is a construct

$$\mathcal{A} = (V, A_1, A_2, \ldots, A_n, K),$$

where

- V is the input alphabet,
- $A_i = (Q_i, V, f_i, q_i, F_i), 1 \leq i \leq n$, are finite automata with the set of states Q_i, $q_i \in Q_i$ (the initial state of the automaton i), $F_i \subseteq Q_i$ (the set of final states), and f_i is the transition mapping of the automaton i defined as follows

$$f_i : Q_i \times V \cup \{\varepsilon\} \longrightarrow 2^{Q_i}.$$

- $K = \{K_1, K_2, \ldots, K_n\} \subseteq \bigcup_{i=1}^{n} Q_i$ is the set of querry states.

The automata $A_1, A_2, \ldots, A_n$ are called the *components* of the system $\mathcal{A}$. If there exists just one $1 \leq i \leq n$ such that $K \subseteq Q_i$, then the system is said to be *centralized*, the master of this system being the component i. For sake of simplicity, whenever a system is centralized, the first component is its master. If the following conditions

$$\begin{array}{ll} (i) & card(f_i(s,a)) \leq 1 \text{ for all } s \in Q_i \text{ and } a \in V \cup \{\varepsilon\}, \\ (ii) & \text{if } card(f_i(s,\varepsilon)) \neq 0 \text{ for some } s \in Q_i, \text{ then} \\ & card(f_i(s,a)) = 0 \text{ for all } a \in V, \end{array}$$

are fulfilled for all $1 \leq i \leq n$, then the automata system is *deterministic*.

By a configuration of a parallel communicating automata system as above, we mean an $2n$-tuple

$$(s_1, x_1, s_2, x_2, \ldots, s_n, x_n)$$

where

- s_i is the current state of the component i,
- x_i is the remaining part of the input word which has not been read yet by the component i, $1 \leq i \leq n$.

We define two binary relations on the set of all configurations of $\mathcal{A}$ in the following way:

$$(s_1, x_1, s_2, x_2, \ldots, s_n, x_n) \vdash (p_1, y_1, p_2, y_2, \ldots, p_n, y_n)$$

iff one of the following two conditions holds:

$$\begin{array}{ll} (i) & K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset \text{ and} \\ & x_i = a_i y_i, a_i \in V \cup \{\varepsilon\},\ p_i \in f_i(s_i, a_i),\ 1 \leq i \leq n \\ (ii) & \text{for all } 1 \leq i \leq n \text{ such that } s_i = K_{j_i} \text{ and } s_{j_i} \notin K \text{ put } p_i = s_{j_i}, \\ & p_r = s_r, \text{ for all the other } 1 \leq r \leq n, \text{ and } y_t = x_t,\ 1 \leq t \leq n. \end{array}$$

$$(s_1, x_1, s_2, x_2, \ldots, s_n, x_n) \vdash_r (p_1, y_1, p_2, y_2, \ldots, p_n, y_n)$$

iff one of the following two conditions holds:

$$\begin{array}{ll} (i) & K \cap \{s_1, s_2, \ldots, s_n\} = \emptyset \text{ and} \\ & x_i = a_i y_i, a_i \in V \cup \{\varepsilon\},\ p_i \in f_i(s_i, a_i),\ 1 \leq i \leq n \\ (ii) & \text{for all } 1 \leq i \leq n \text{ such that } s_i = K_{j_i} \text{ and } s_{j_i} \notin K \text{ put } p_i = s_{j_i},\ p_{j_i} = q_{j_i}, \\ & p_r = s_r, \text{ for all the other } 1 \leq r \leq n, \text{ and } y_t = x_t,\ 1 \leq t \leq n. \end{array}$$

The difference between the two relations defined above may be easily noticed when the current states of some components are querry states: these components get into communication with those components identified by the query states, which are forced to send their current states, providing they are not query states, these states becoming the new states of the receiver components. The next states of the sender components remain the same in the case of relation $\vdash$ whereas they become the initial states when relation $\vdash_r$ has been applied.

A parallel communicating automata system whose all moves are based on the relation $\vdash_r$ is said to be *returning.*

Informally, the language accepted by a PCFAS $\mathcal{A}$, consists of all strings $x \in V^*$ such that the system starts in an initial configuration $(q_1, x, q_2, x, \ldots, q_n, x)$ and reaches a final configuration, that is a configuration of the form $(s_1, \varepsilon, s_2, \varepsilon, \ldots, s_n, \varepsilon)$, with $s_i \in F_i$. Formally

$$\begin{aligned} Rec(\mathcal{A}) &= \{x \in V^* | (q_1, x, q_2, x, \ldots, q_n, x) \vdash^* (s_1, \varepsilon, s_2, \varepsilon, \ldots, s_n, \varepsilon), \\ &\quad s_i \in F_i, 1 \le i \le n\}, \\ Rec_r(\mathcal{A}) &= \{x \in V^* | (q_1, x, q_2, x, \ldots, q_n, x) \vdash_r^* (s_1, \varepsilon, s_2, \varepsilon, \ldots, s_n, \varepsilon), \\ &\quad s_i \in F_i, 1 \le i \le n\}. \end{aligned}$$

We shall denote by:

- *rcpcfa*(n) - a returning centralized parallel communicating finite automata system of degree n;

- *rpcfa*(n) - a returning parallel communicating finite automata system of degree n;

- *cpcfa*(n) - a centralized parallel communicating finite automata system of degree n;

- *pcfa*(n) - a parallel communicating finite automata system of degree n.

We add the prefix d in order to denote deterministic variants. If $x(n)$ is a type of automata system, then $X(n)$ is the class of all languages accepted by automata systems of type x. For example, *RCPCFA*(n) is the class of all languages accepted by *rcpcfa*(n) automata systems.

Example 1. *Consider the cpcfa*(2)

$$\mathcal{A} = (\{a, b, c\}, A_1, A_2, \{K_1, K_2\}),$$

where A_1 and A_2 are two deterministic finite automata with ε-moves whose transition mappings are listed below

$$\begin{array}{ll} f_1(q_1, \varepsilon) = K_2 & f_2(q_2, a) = q_2, \\ f_1(s_1, a) = K_2 & f_2(q_2, b) = s_1, \\ f_1(q_2, \varepsilon) = K_2 & f_2(s_1, b) = s_1, \\ f_1(s_2, b) = K_2 & f_2(s_1, c) = s_2, \\ f_1(s_f, c) = q_f & f_2(s_2, c) = s_2, \\ f_1(q_f, c) = q_f & f_2(s_2, \varepsilon) = s_f, \\ & f_2(s_f, \varepsilon) = s_f. \end{array}$$

By taking the sets of final states as being $F_1 = \{q_f\}$ and $F_2 = \{s_f\}$ we get $L(\mathcal{A}) = \{a^nb^nc^n|n \geq 1\}$, which is a non-context-free language.

Note that, directly from definitions it follows:

Lemma 1. *1. $RCPCFA(n) \subseteq RPCFA(n)$ and $CPCFA(n) \subseteq PCFA(n)$, for all $n \geq 1$. Moreover, every family $X(1)$, $X \in \{RCPCFA, RPCFA, CPCFA, PCFA\}$, equals the family of regular languages.*

2. $X(n) \subseteq X(n+1)$ for all $X \in \{RCPCFA, RPCFA, CPCFA, PCFA\}$.

3. All the above relations hold for deterministic variants as well.

3 Computational power

In the first part of this section we compare the computational power of the automata systems previously introduced with the computational power of multi-head finite automata. For technical reasons, we shall use here the following definition of multi-head finite automata.
A (nondeterministic) k-head finite automaton is a quintuple

$$A = (k, Q, V, f, q_0, F),$$

where Q, V, q_0, F have the same meaning as for a usual finite automaton, and f is a mapping from $Q \times (V \cup \{\varepsilon\})^k$ into the subsetes of Q. The above definition is essentially similar to that found in [11] and [13]. Thus, $q \in f(s, a_1, a_2, \ldots, a_k)$ indicates that the automaton in state s each head i reading a_i may enter state q. The input heads are idealized in the sense that they may pass over one another freely and they are prevented from going off the right end of the input. Moreover, if a head reads ε, it does not move to the right and if it reads a symbol in V, it moves to the right one square. Acceptance is defined as follows: a string is accepted if the automaton starts in the initial state with the string on the input tape, all heads being positioned on the leftmost symbol of the input, and enters, after finitely many moves, in a final state, the input being completely read by all heads. In all the other cases, the input string is rejected. For a multi-head finite automaton A as above denote by $Rec(A)$ the set of all strings accepted by A.

Theorem 1 [18] *1. $X(n)$ is included in the class of lanaguages accepted by n-head finite automata for all $X \in \{RCPCFA, RPCFA, CPCFA, PCFA\}$.*

2. A language is accepted by a n-head finite automaton if and only if it belongs to $PCFA(n)$.

3. $X(n)$ is included in the class of lanaguages accepted by deterministic n-head finite automata for all $X \in \{DRCPCFA, DRPCFA, DCPCFA, DPCFA\}$.

4. A language is accepted by a deterministic n-head finite automaton if and only if it belongs to $DPCFA(n)$.

In the sequel we define two operations on words and languages useful in our considerations. A homomorphism which erases some symbols and leaves unchanged the others is said to be a *projection*. A projection $h : (V \cup V')^* \longrightarrow V^*$ that erases the symbols in V' only is denoted by pr_V. The other operation is a wellknown operation in formal language theory and in parallel programming theory, called *shuffle*. A shuffle of two strings is an arbitrary interleaving of the substrings of the original strings, like shuffling two decks of cards. More precisely, for two strings $x, y \in V^*$ and two symbols $a, b \in V$,

$$\begin{array}{ll} (i) & x \sqcup\!\sqcup \varepsilon = \varepsilon \sqcup\!\sqcup x = x, \\ (ii) & ax \sqcup\!\sqcup by = a(x \sqcup\!\sqcup by) \cup b(ax \sqcup\!\sqcup y). \end{array}$$

For two languages L_1, L_2 we define

$$L_1 \sqcup\!\sqcup L_2 = \bigcup_{x \in L_1, y \in L_2} x \sqcup\!\sqcup y.$$

It is known the following representation of recursively enumerable languages, see, e.g., [9]:

Theorem 2. *Each recursively enumerable language $L \subseteq T^*$ can be written as $L = pr_T(TS(V) \cap R)$, where V is an alphabet including T, R is a regular language and $TS(V)$ is the twin shuffle language over the alphabet V defined by $TS(V) = \bigcup_{x \in V^*} x \sqcup\!\sqcup \bar{x}$.*

Based on this result, the following characterizations of the recursively enumerable languages class is given in [18].

Theorem 3. *1. A language $L \subseteq T^*$ is recursively enumerable if and only if $L = pr_T(Rec_r(\mathcal{A}))$, where $\mathcal{A}$ is an automata system in $\{rcpcfa(3), rpcfa(3)\}$.*

2. A language $L \subseteq T^*$ is recursively enumerable iff $L = pr_T(Rec(\mathcal{A}))$, where $\mathcal{A}$ is a $cpcfa(3)$.

3. A language $L \subseteq T^*$ is recursively enumerable iff $L = pr_T(Rec(\mathcal{A}))$, where $\mathcal{A}$ is a $pcfa(2)$.

4 Some undecidable problems

In this section we investigate the decidability status of some "classical" problems for the automata systems previously introduced. One of the most important matters is the membership problem. This problem is decidable in polynomial time since all automata systems defined in the previous section can be effectively simulated by multi-head finite automata, as Theorem 1 claims, for which the membership problem is polynomially time solvable [3, 23].

In the sequel, we shall consider the equivalence and inclusion problems which will turn out to be undecidable via the undecidability of other problems. We start with some lemmata which the undecidability results are based on; we prefer to recall them because they prove also some closure properties of the language families defined by finite automata systems.

Lemma 2.[16] *1. Both families $CPCFA$ and $PCFA$ are effectively closed under union.*

2. Both families RPCFA and PCFA are effectively closed under intersection.

The universe problem for an automata system $\mathcal{A}$ over an alphabet V asks whether $\mathcal{A}$ accepts all words in V^*. The emptiness problem asks whether or not the system rejects all words. Now we are able to prove the main result of this paper.

Theorem 4.[16] *1. The universe problem is undecidable for pcfa(n)'s and cpcfa(n)'s for all $n \geq 5$.*

2. The emptiness problem is undecidable for pcfa(n)'s and rpcfa(n)'s for all $n \geq 4$.

Based on these theorems, and a bit different reduction to the Post Correspondence Problem the next result is proved in [16]:

Theorem 5. *The equivalence and inclusion problems are undecidable for pcfa(n)'s, rpcfa(n)'s and cpcfa(k)'s, for all $n \geq 4$ and $k \geq 5$.*

As any $pcfa(n)$ can be effectively simulated by a n-head finite automaton (see Theorem 1) one may easily infer that the problems considered here are decidable neither for n-head automata for all $n \geq 4$.

Along the same lines there are plenty of questions whose answers are not known by us. We list here some of them.

1. Which of the problems investigated in this paper are decidable for $rpcfa(k)$'s and $pcfa(k)$'s for $k = 2, 3$?
2. The same question for $cpcfa(k)$'s for $k = 2, 3, 4$?
3. What one can say about the decidability status of these problems for $rcpcfa$'s or for the deterministic variants of all systems considered here?

5 Automata systems as mildly context-sensitive acceptors

This part is dedicated to a brief discussion regarding a possible connection between the classes of languages accepted by the parallel communicating automata systems defined in this paper and the so-called mildly context-sensitive class [15]. This class contain the context-free languages, covers the specific non-context-free constructions in natural languages, but are parsable in polynomial time and have the "bounded growth property" (the length difference between two words, so that no word of an intermediate length is in the language, is bounded). The last property is sometimes replaced by semilinearity which is a much stronger property.

The reason that all classes $RCPCFA(k), RPCFA(k), CPCFA(k), PCFA(k)$ contains only semilinear languages is that Ibarra showed that the languages accepted by multihead nondeterministic pushdown automata satisfy the semilinearity property [14]. By Theorem 1 this property is extended to all classes mentioned above.

It is known that every nondeterministic k-head pushdown automata language can be recognized by a deterministic Turing machine in $n^{2.81k}$ time, see, e.g., [23] or [3]. By the translation provided by Theorem 1 we conclude that all classes defined here are polynomially parsable.

As far as some specific non-context-free constructions in natural languages the reader is refered to [10] and [17] for a detailed discussion. We present here three of them which linguists seemed to finally agree on: replication (modelled by the formal language $L_1 = \{x\#x \mid x \in \{a,b\}^+\}$), multiple agreements (ilustrated by $L_2 = \{a^n b^n c^n \mid n \geq 1\}$), and crossed dependencies (as in $L_3 = \{a^n b^m c^n d^m \mid n, m \geq 1\}$). The next result shows that parallel communicating finite automata systems having just two components are able to recognize the the languages L_1, L_2, L_3 under all strategies of communication.

Proposition 1. *1. The languages L_2, L_3 lie in any of the families $RCPCFA(2)$, $CPCFA(2)$, $RPCFA(2)$, $PCFA(2)$.*

2. The languages L_1 lies in any of the families $RCPCFA(2)$, $CPCFA(3)$, $RPCFA(2)$, $PCFA(2)$.

We point out some natural questions refering the results in the previous section.

1. Do the first two items of Theorem 3 remain still valid for automata systems with two components?

2. By Theorem 1 and [24] we infer that $n+1$ components can do more than n components in a *pcfa*. In the aforementioned paper, Yao and Rivest consider the languages

$$L_m = \{w_1\#w_2\#\dots w_{2m} \mid w_i \in \{a,b\}^*, \text{ and } w_i = w_{2m-i+1}, 1 \leq i \leq m\}.$$

They show that $L_{\binom{k}{2}}$ can be recognized by a k–head finite automaton but there is no $k-1$–head finite automaton able to recognize it. Each language L_m is in both families $RCPCFA(m+1)$ and $RPCFA(m+1)$, consequently

$$X(k-1) \subset X\left(\binom{k}{2} + 1\right),$$

holds for $X \in \{RCPCFA, RPCFA\}$. However we were not able to prove a hierarchy result

$$X(k) \subset X(k+l)$$

for some constant l. Does such a hierarchy exist?

6 The degree of communication

In this section, we define a dynamical measure of descriptional complexity for parallel communicating finite automata systems, following [19]. This criterion appears to be quite appropriate for investigating some complexity aspects of these automata systems.

As we have seen, a configuration may contain several query states which can be satisfied in one or more communication steps. For each accepting/communication step we count the number of query states satisfied in that step. Thus, for a given configuration $(q_1, x_1, q_2, x_2, \dots, q_n, x_n)$ we define the number

$$C(q_1, q_2, \dots, q_n) = \mathrm{card}(\{q_i \mid q_i = K_{j_i} \text{ for some } 1 \le j_i \le n \text{ and } q_{j_i} \notin K, 1 \le i \le n\}.$$

Let $\mathcal{A} = (V, A_1, A_2, \dots, A_n, K)$ be a parallel communicating finite automata system and P be a path in $\mathcal{A}$:

$$P: \ (q_1, x, q_2, x, \dots, q_n, x) \vdash (q_1^1, x_1^1, q_2^1, x_2^1, \dots, q_n^1, x_n^1) \vdash \dots \vdash (q_1^k, \varepsilon, q_2^k, \varepsilon, \dots, q_n^k, \varepsilon),$$

for some $k \ge 1$ and $q_i^k \in F_i, 1 \le i \le n$. We define the degree of communication of $\mathcal{A}$ for x in P as the number

$$Comm(x, P) = \sum_{i=1}^{k} C(q_1^i, q_2^i, \dots, q_n^i).$$

Moreover, the degree of communication of $\mathcal{A}$ for x is defined by

$$Comm(x) = \min\{Comm(x, P) \mid P \text{ is an accepting path for } x \text{ in } \mathcal{A}\}.$$

The degree of communication of $\mathcal{A}$ is

$$Comm(\mathcal{A}) = \sup\{Comm(x) \mid x \in Rec(\mathcal{A})\}.$$

For a language L and a class X of parallel communicating finite automata systems, $X \in \{rcpcfa, cpcfa, rpcfa, pcfa\}$, we define

$$Comm_X(L) = \inf\{Comm(\mathcal{A}) \mid L = Rec(\mathcal{A})\}.$$

Example 2 *Let us consider again the $cpcfa(2)$ given in Example 1. For each word $x = a^k b^k c^k$ in $Rec(\mathcal{A})$ we have $Comm(x) = 2k$. Consequently, $Comm(\mathcal{A}) = \infty$. However, $Comm(Rec(\mathcal{A})) = 0$ as shown by the following deterministic $cpcfa(3)$:*

$$\begin{array}{lll}
f_1(q_1, \varepsilon) = s_1 & f_2(q_2, a) = r_1 & f_3(q_3, a) = t_1, \\
f_1(s_1, a) = s_2 & f_2(r_1, a) = r_1 & f_3(t_1, a) = t_1, \\
f_1(s_2, \varepsilon) = s_1 & f_2(r_1, \varepsilon) = r_2 & f_3(t_1, b) = t_2, \\
f_1(s_1, b) = s_3 & f_2(r_2, b) = r_3 & f_3(t_2, b) = t_2, \\
f_1(s_3, b) = s_3 & f_2(r_3, \varepsilon) = r_2 & f_3(t_2, \varepsilon) = t_3, \\
f_1(s_3, c) = s_4 & f_2(r_2, c) = r_4 & f_3(t_3, c) = t_4, \\
f_1(s_4, c) = s_4 & f_2(r_4, c) = r_4 & f_3(t_4, \varepsilon) = t_3,
\end{array}$$

with $F_1 = \{s_4\}, F_2 = \{r_4\}, F_3 = \{t_4\}$. One can easily check that this automata system recognizes the language $\{a^n b^n c^n \mid n \geq 1\}$. Indeed, the system recognizes words of the form $a^i b^j c^k$ only, with

$$2i + j + k = i + 2j + k = i + j + 2k$$

that is $i = j = k$. Moreover, the accepting process for each word requires no communication step.

The next result is based on a simple construction involving multi-head automata.

Theorem 6 *Let $\mathcal{A}$ be an automata system of any type in $\{rcpcfa, cpcfa, rpcfa, pcfa\}$, and x be a word recognized by $\mathcal{A}$. Then, one can compute $Comm(x)$.*

A natural problem concerns the computability of this measure for an automata system and for a language accepted by a given automata system. We cannot provide here a complete answer to this problem.

Theorem 7 *$Comm(Rec(\mathcal{A}))$ cannot be algorithmically computed for an arbitrarily given automata system $\mathcal{A}$ of type cpcfa or pcfa.*

We believe that the degree of communication can be algorithmically computed neither for automata systems of any type nor for the languages recognized by them.

There are plenty of open problems; we list here some of those that appear quite attractive to us:

1. Given an arbitrary positive integer n, do a language L and an automata system of type X exist such that $Comm_X(L) = n$?

2. Can we compute the degree of communication of any of the four variants of automata systems?

3. What can one say about the computability status of the degree of communication for languages recognized by the other two variants of automata systems?

References

[1] A. Atanasiu, V. Mitrana, The modular grammars, *Intern. J. of Computer Mathematics* 30 (1989), 17–35.

[2] A. O. Buda, Multiprocessor automata, *Inform. Proces. Lett.* 25 (1977), 257–261.

[3] S. A. Cook, Characterizations of pushdown machines in terms of time - bounded computers, *Journal of ACM* 18 (1971), 4–18.

[4] E. Csuhaj-Varjú, J. Dassow, On cooperating/distributed grammar systems, *J. of Information Processing and Cybernetics (EIK)*, 26 (1990), 49–63.

[5] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A grammatical approach to distribution and cooperation.* Gordon and Breach, 1994.

[6] E. Csuhaj-Varju, C. Martín-Vide, V. Mitrana, G. Vaszil, Parallel communicating pushdown automata systems, *Intern. J. Found. Comp. Sci.*, in press.

[7] J. Dassow, V. Mitrana, Stack cooperation in multi-stack pushdown automata. *J. Comput. System Sci.* 58 (1999), 611–621.

[8] E. H. Durfee et all, Cooperative distributed problem solving. in *The Handbook of AI*, vol. 4 (A. Barr, P. R. Cohen, E. A. Feigenbaum eds.), Addison-Wesley, Reading Mass., 1989.

[9] J. Engelfriet, G. Rozenberg, Fixed point languages, equality languages, and representations of recursively enumerable languages, *J. of ACM* 27 (1980), 499–518.

[10] G. Gazdar, G. K. Pullum, Computationally relevant properties of natural languages and their grammars, *New Generation Computing* 3 (1985), 273–306.

[11] J. Hartmanis, On nondeterminancy in simple computing devices, *Acta Informatica* 1 (1972), 336–344.

[12] M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley Publ. Co., 1978.

[13] O. H. Ibarra, One two-way multihead automata, *J. Comput. System Sci.* 7 (1973), 28–36.

[14] O. H. Ibarra, A note on semilinear sets and bounded-reversal multihead pushdown automata, *Inform. Process Lett.* 3 (1974), 25–28.

[15] A. K. Joshi, How much context-sensitivity is required to provide reasonable structural descriptions? Tree adjoining grammars. In *Natural Language Processing: Psycholinguistic, Computational and Theoretic Perspectives* (D. R. Dowty et al., eds.), Cambridge Univ. Press, New York, 1985, 206–250.

[16] C. Martín-Vide, V. Mitrana, Some undecidable problems for parallel communicating finite automata systems, submitted.

[17] B. H. Partee, A. ter Meulen, R. E. Wahl, *Mathematical Methods in Linguistics*, Kluwer Academic Publ., Dordrecht, Boston, London, 1990.

[18] A. Mateescu, V. Mitrana, A. Salomaa, Parallel finite automata systems communicating by states, submitted.

[19] V. Mitrana, The degree of communication in parallel communicating finite automata systems, *Journal of Automata, Languages and Combinatorics*, in press.

[20] Gh. Păun, L. Sântean, Parallel communicating grammar systems: the regular case, *Ann. Univ. Bucharest, Ser. Matem.-Inform.* 38 (1989), 55–63.

[21] G. Păun, G. Rozenberg, A. Salomaa, *DNA Computing. New Computing Paradigms*, Springer-Verlag, 1998.

[22] G. Rozenberg, A. Salomaa (eds.), *The Handbook of Formal Languages*, Springer-Verlag, 1997.

[23] I. H. Sudborough, Some remarks on multihead automata, *R.A.I.R.O. Informatique théorique* 11 (1977), 181–195.

[24] A. C. Yao, R. L. Rivest, k+1 heads are better than k, *Journal of ACM* 25 (1978), 337–340.