Communication Gap for Finite Memory Devices^{*}

Tomasz Jurdziński^{* †} and Mirosław Kutylowski^{‡ §}

^{*} Institute of Computer Science, Wrocław University

[†]Department of Computer Science, Technical University of Chemnitz

[‡]Department of Mathematics and Computer Science, Poznań University

[§]Institute of Mathematics, Wrocław University of Technology

Abstract. So far, not much is known on communication issues for computations on distributed systems, where the components are weak and simultaneously the communication bandwidth is severely limited. We consider synchronous systems consisting of finite automata which communicate by sending messages while working on a shared read-only data. We consider the number of messages necessary to recognize a language as a its complexity measure.

We present an interesting phenomenon that the systems described require either a constant number of messages or at least $\Omega((\log \log \log n)^c)$ of them (in the worst case) for input data of length n and some constant c . Thus, in the hierarchy of message complexity classes there is a gap between the languages requiring only $O(1)$ messages and those that need a non-constant number of messages. We show a similar result for systems of one-way automata. In this case, there is no language that requires $\omega(1)$ and $o(\log n)$ messages (in the worst case). These results hold for any fixed number of automata in the system.

The lower bound arguments presented in this paper depend very much on results concerning solving systems of diophantine equations and inequalities.

Track A, keywords: automata, computational complexity, parallel and distributed processing

1 Introduction

Traditional complexity theory usually disregards either communication between processing units or their internal limitations. On the other hands, for many reasons it is interesting to consider the situation where we have many weak processing units and severely limited communication between them. Not much is known about communication complexity of problems in this setting. The classical complexity theory concentrates on the situation when input data is divided into parts and every part is available to exactly one participant, while local computational resources are unlimited. Perhaps, the major achievement for restricted local resources are the results of Beame et al. [1] on tradeoffs between the communication volume and space used.

We consider a setting, in which we do not use partitioning of input data for getting lower bounds. While the whole input is available to every processing unit we show that internal limitations of processing units cause large communication requirements.

^{*} This research was partially supported by Deutsche Forschungsgemeinschaft and Komitet Badań Naukowych, grants 8 T11C 032 15, 8 T11C 012 18, and finally 7 T11C 032 20. Some of the results are contained in the Ph.D. dissertation of the first author. Some work has been done when the second author was visiting University of Mannheim.

Model. We consider the simplest model one can imagine to study the problem mentioned above, namely systems of (a finite number of) finite automata (equivalent to “multihead finite automata” [5], see also [12]). The automata work independently on a common read-only input data and communicate by sending (broadcasting) messages. Since each automaton has only finitely many internal states, the number of different messages, as well as their size, is constant with respect to the size of the data. So the number of messages corresponds roughly to communication volume. The transition function of a single automaton depends on its internal state, an input symbol seen and the messages received from other automata at this moment. The automata work in parallel and synchronously, the messages are delivered instantly. More formally:

Definition 1. We say that $M = (A_1, A_2, \dots, A_k)$ is a system of finite automata, if there is a finite set Q such that every internal state of any A_i is in Q ; each automaton has a single read-only head; and a step of M is executed as follows: if A_i , for $i \in \{1, \dots, k\}$, is in a state q and a is an input symbol seen by the read-head of A_i , then:

- (1) A_i broadcasts a message $m = \mu_i(q, a)$ to other automata (for $\mu_i(q, a) = \text{nil}$, no message is sent by A_i);
- (2) A_i changes its internal state to q' and moves its read head r positions to the right ($r \in \{0, 1, -1\}$), where q' and r are given by transition function δ_i :

$$(q', r) = \delta_i(q, a, m_1, \dots, m_{i-1}, m_{i+1}, \dots, m_k),$$

m_j denotes the message sent by automaton A_j during the first phase of the step or “nil”, if no such a message is sent.

Communication configuration is a configuration of the system in which at least one automaton sends a message (and the final configuration – the configuration in which every automaton is in the **final state**, i.e., in the state for which no transition is defined, independently of messages received and symbol observed). An **initial configuration** of the system M is the configuration in which all automata see the leftmost cell of the input tape and are in initial states $q_0^1, \dots, q_0^k$. An input word x is given on the tape together with endmarkers $\triangleright$ and $\triangleleft$ preventing the heads to leave $\triangleright x \triangleleft$. A system accepts an input x when a fixed automaton (for instance the first one) enters an accepting state (which is the final state) during the computation on $\triangleright x \triangleleft$ which starts in the initial configuration. A system recognizes language L , when a fixed automaton accepts input x if and only if $x \in L$.

Message complexity classes. Let $\text{MESSAGE}(f(n))$ denote the class of languages that may be recognized by a system of automata such that at most $f(n)$ communication configurations occur for computation of the system on every input word of length n (observe that the number of messages sent is at most k times greater than the number of communication configurations). Note that already $\text{MESSAGE}(1)$ contains a non-regular language $\{a^m b^m \mid m \in \mathbb{N}\}$ and a non-context-free language $\{a^m b^m c^m \mid m \in \mathbb{N}\}$. For example, to recognize $\{a^m b^m \mid m \in \mathbb{N}\}$ by 2 automata and one message we start both automata moving right in such a way that the automaton A_2 makes two steps on each symbol and the automaton A_1 makes one step on each symbol (so A_2 moves two times faster than A_1). Additionally, A_2 checks if the input has the form $a^* b^*$. Once A_2 arrives

at the right endmarker, it sends the message. The input is accepted if and only if A_1 receives the message just in the moment when it is on the border between a 's and b 's.

A number of hierarchy results concerning message complexity classes have been obtained in [8]. While there is a strict hierarchy concerning a constant number of communication configurations (or messages), for non-constant number of messages, the hierarchy of this kind starts from $\text{MESSAGE}(\log \log n)$. It is also known [8] that the language

$L_2 = \{1^{f_1} \# 1^{f_2} \# \dots \# 1^{f_k} : f_1 = 2, f_2 = 3, f_{i-1} \mid (f_i - 1), f_{i-2} \mid (f_i - 1), f_i > 1 \text{ for } i = 3, \dots, k\}$ is in $\text{MESSAGE}(O(\log \log n))$ but not in any class of the type $\text{MESSAGE}(o(\log \log n))$.

2 New results

We show a gap phenomenon for message complexity classes: if the number of messages is non-constant, it must reach a certain level. Results of this kind are not sparse in complexity theory, but they are a little bit surprising in the context of communication complexity (note that for the classical communication complexity it does not occur). Our proof method relies on reducing the problems to questions about the size of minimal solutions of some Diophantine systems. In references in [11] one can find more examples of reductions of complexity theory problems to “Diophantine problems”.

For one-way automata (for which the heads cannot move to the left) we show that there is a gap between $\Theta(1)$ and $\Theta(\log n)$ message complexity classes:

Theorem 1. *For a function $f(n)$ such that $f(n) = \omega(1)$ and $f(n) = o(\log n)$, there is no one-way multi-automata system which requires $\Theta(f(n))$ communication configurations.*

The bound $\log n$ is optimal, since it is easy to see that the language $L_1 = \{1^{2^0} \# 1^{2^1} \# 1^{2^2} \# \dots \# 1^{2^k} : k \in \mathbb{N}\}$ can be recognized with $O(\log n)$ messages: (the algorithm is based on the trick with different “speeds” presented above for the language $\{a^m b^m \mid m \in \mathbb{N}\}$). For two-way systems, we have a much smaller bound (note that it may not exceed $\log \log n$, as shown by the language L_2).

Theorem 2. *There is a constant c such that for $f(n) = \omega(1)$ and $f(n) = o((\log \log \log n)^c)$, there is no two-way multi-automata system which requires $\Theta(f(n))$ communication configurations.*

Outline of the proofs. Let S be a system of finite automata that needs $\omega(1)$ communication configurations. Our goal is to show that if there is an input that requires exactly g communication configurations, then there is an input of length $O(2^g)$ (in the case of one-way systems) or $2^{2^{u(g)}}$ for some polynomial u (in the case of two-way systems) with a similar history of computation, and, what is most important, for which g communication configurations occur. Then, Theorems 1 and 2 follow immediately.

The first step in finding such inputs is to show that after dividing computations into some number of categories, each category can be described by a diophantine system. Such a system consists of linear equations and inequalities with integer coefficients with absolute values bounded by a constant (in the the case of one-way automata) and,

additionally, of expressions like $f(x)|g(x)$, where f and g are linear functions with integer coefficients bounded by a constant (in the general case).

The most important feature of diophantine systems derived is that any computation corresponds to some diophantine solution of the system (i.e., a solution in $\mathbb{N}$), and vice versa, every diophantine solution describes some input data. Additionally, the input length is bounded by a linear expression in some of the variables occurring in the diophantine system.

To complete the proof, we argue that there is a small solution of the diophantine system. Such a solution corresponds to an input word, for which, in turn, the number g of communication configurations is relatively big, yielding the claimed bound on the number of communication configurations in the worst case. For linear equations and inequalities with coefficients having absolute values bounded by a constant the minimal solution is at most exponential in the size of the system [4] (see [3] for efficient methods for finding minimal solutions). For the case of divisibilities, one may analyze the construction of Lipshitz [9] to find that the minimal solution is at most triple exponential.

The key technical part of this paper (Section 3) is expressing behavior of systems of automata by diophantine systems. In Section 4, we guide over the results concerning the size of minimal solutions of diophantine systems.

3 Reduction to diophantine problems

First, we introduce some notions describing work of automata during periods when no messages are sent. Let $\{A_i\}_{i=1,\dots,k}$ be a set of finite automata of the system S , let Q be the set of their states. **Transition** is a tuple (A_i, q, s, q', s') where $q, q' \in Q$ and $s, s' \in \{Left, Right\}$. Word w satisfies a transition (A_i, q, s, q', s') , if automaton A_i , starting the computation in state q on the side s of w , leaves the word w in state q' on the side s' (if it receives no message in this time) and does not send any message in this part of computation. **Clocked transition** is a tuple (A_i, q, s, q', s', t) where $q, q' \in Q$, $s, s' \in \{Left, Right\}$ and $t \in \mathbb{N}$. Word w satisfies a clocked transition (A_i, q, s, q', s', t) , if A_i makes t steps in part of computation which starts when it enters w in the state q on the side s and finishes when it leaves w in the state q' on the side s' (and A_i does not leave w between these two events). **Transition set** is a set of transitions. **Clock characterization** is a set of clocked transitions. A **communication partition** of a word x by the system S is a sequence of words $x_1, \dots, x_{h+1}$ and a sequence of symbols $\sigma_1, \dots, \sigma_h$ such that $x = x_1\sigma_1x_2\sigma_2\dots x_h\sigma_hx_{h+1}$ and:

1. during computation on x , at each communication configuration and at the final configuration of S the read-heads occupy positions from the set $\{|x_1\sigma_1\dots x_i\sigma_i| : i \in \{1, \dots, h\}\}$
2. for each $i \in \{1, \dots, h\}$, there is a communication configuration, in which some head scans σ_i (i.e., it is on position $|x_1\sigma_1\dots x_i\sigma_i|$).

The positions $u = |x_1\sigma_1\dots x_i\sigma_i|$ for $i \in \{1, \dots, h\}$, are called **communication positions**. Obviously, $h \leq k \cdot (g + 1)$, where k is the number of automata and g the number of communication configurations. The words $x_1, x_2, \dots, x_{h+1}$ are called **silent blocks** of x .

A **communication pattern** of the system S on input x , is a set of tuples of the form $\langle i, (q_1, i_1), (q_2, i_2), \dots, (q_k, i_k) \rangle$ such that when the i th communication configuration occurs the automaton A_i is in the state q_l on position $|x_1 \sigma_1 \dots x_{i_l} \sigma_{i_l}|$ for $l \leq k$. By a **full transition set** of the block x_i of the input word x we mean a transition set of x_i which consists of all transitions on x_i made during the computation of the system S on x . For any word w satisfying the full transition set T , a **full clock characterization** of w is its clock characterization consisting of its clocked transitions of all transitions from T .

A **computation digest** of the system S on word x , denoted by $H_S(x)$ consists of:

- a sequence $\sigma_1, \dots, \sigma_h$ of symbols on consecutive communication positions of x ,
- a *communication pattern* of S on x ,
- *full transition sets* of all silent blocks of x .

Observe that a computation digest fixes neither the silent blocks $x_1, \dots, x_h$ nor their full clock characterizations.

A state q is the **blocking state** for an automaton A_i and a symbol a if A_i makes no move and no change of its state when it is in state q , reads input symbol a and no message arrives, i.e., $(q, 0) = \delta_i(q, a, nil, \dots, nil)$,

3.1 Outline of diophantine characterization of one-way automata

First, we give an idea of a “linear” characterization of all inputs with computation digest $H_S(x) = H$. Inside a silent block x_k of x no message is sent or received. So, from the point of view of computation of the system, the only important property of x_k is its full clock characterization, C_k . If any other word x'_k has the same full clock characterization C_k , then we may replace x_k with x'_k without changing messages sent (and acceptance). If x'_k is shorter than x_k , then we obtain a shorter input with computation digest H . However, in most cases it would be impossible to shorten one block only, without changing the computation digest. Our approach is to describe dependencies between the blocks and then to shorten the blocks simultaneously keeping these dependencies valid.

Example. To see how characterizations work, we consider a system S with two automata A_1 and A_2 on the word x that has a communication partition which consists of symbols $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ and silent blocks x_1, x_2, x_3, x_4 . The automaton A_1 (respectively A_2) visits communication positions in states q_1, q_2, q_3, q_4 (respectively q'_1, q'_2, q'_3, q'_4). The *communication pattern* of S on x equals $\{ \langle 1, (q_2, 2), (q'_1, 1) \rangle, \langle 2, (q_3, 3), (q'_2, 2) \rangle, \langle 3, (q_4, 4), (q'_4, 4) \rangle \}$. Assume also that automaton A_1 (respectively A_2) sends messages only in states q_2 and q_4 (respectively q'_2), and q_3 is the only blocking state. Let $t_{1i}, \dots, t_{4i}$ denote the numbers of steps spent by automaton A_i on blocks $x_1, \dots, x_4$ for $i \in \{1, 2\}$. We may express H , the computation digest of S on x , by full transition sets of blocks x_1, x_2, x_3, x_4 and the conditions:

$$t_{11} + t_{21} = t_{12}, \quad t_{22} \geq t_{31}, \quad t_{41} = t_{32} + t_{42}.$$

In general, once we have equations and inequalities as in the example above, we look for the words $x'_1, \dots, x'_h$ which have :

- *full transition sets* determined by the computation digest H ,
- *full clock characterizations* satisfying an appropriate system of linear equations and inequalities (determined by the *communication pattern*).

The equations and inequalities mentioned above describe dependencies between the numbers of steps made by the automata when they go through silent blocks. However, the variables denoting the numbers of steps made by different automata on the same block (like t_{11} and t_{12} in the example), are dependent. The main difficulty here is that the “speed” of each automaton within a block fluctuates heavily depending on the input symbols read. “Speed pattern” might be extremely complicated. However, going through different speed phases is like walking through a graph of different “speed states”. This graph has a constant size, depending on automata only. So, we may hope to describe such walks by linear expressions with variables describing how many times one goes through certain cycles in the graph.

In fact, implementing this idea is nontrivial. To the full transition set of the block x_i we assign not a single set of expressions and inequalities but a family L_i of such sets. Each set $\phi \in L_i$ consists of some linear inequalities and a linear expression l_{ij} for each automaton A_j . The expression l_{ij} computes t_{ij} , the number of steps made by the automaton A_j on block i of the communication partition.

L_i describes all words with a given full transition set in the following sense: Each such a word corresponds to an assignment of variables for some $\phi \in L$ that fulfills the conditions in ϕ . Vice versa: for each set ψ of L each assignment of variables that satisfies conditions of ψ corresponds to a computation on some word. It is important for us that the length of the silent block i is upper bounded by each expression l_{ij} .

3.2 Linear Characterization Lemma for one-way automata

Lemma 1 (Linear Characterizations Lemma). *Let S be a system of k one-way automata, let H be a computation digest of S on some input x and g communication configurations occur during a computation described by H . We can associate with H a family L of systems of linear diophantine equations and inequalities such that for any system of equations $Ay = b$ and inequalities $Cy \geq d$ in L the following conditions hold:*

1. *the number of equations and inequalities occurring in the system is $O(g)$,*
2. *the number of variables of the system is $O(g)$,*
3. *coefficients of A, b, C and d are integers with absolute values limited by some constant c_2 which depends only on k and the number of the states of automata,*
4. *let $x' = x'_1\sigma_1 \dots x'_h\sigma_h$ be any word such that its computation digest is H , communication partition consists of symbols $\sigma_1, \dots, \sigma_h$ and blocks $x'_1, \dots, x'_h$. Let*

$$\{\langle A_i, q_{ij}, s_{ij}, q'_{ij}, s'_{ij}, t_{ij}(x') \rangle \mid 1 \leq i \leq k\}$$

denote the full clock characterization of word x'_j ¹ (so $t_{ij}(x')$ denotes the number of steps made by A_i on x'_j , the j -th silent block). Then, there are $\gamma_1, \dots, \gamma_m \in \mathbb{N}$, which form a solution of some system in L such that $t_{ij}(x') = \gamma_{(j-1)k+i}$ for $1 \leq i \leq k$, $1 \leq j \leq h$. Conversely, for each solution $\gamma_1, \dots, \gamma_m$ of any system in L one can find a word w with computation digest H such that $t_{ij}(w) = \gamma_{(j-1)k+i}$ for $1 \leq i \leq k$, $1 \leq j \leq h$ (note that m may be bigger than kh).

¹ there is only one tuple for every automaton, since we consider one-way devices

The rest of this section is devoted to the proof of Linear Characterizations Lemma. The key part is to describe dependencies between number of steps necessary to read the same silent block by different automata. We achieve this in the following lemma:

Lemma 2. *Let $T = \{\langle A_i, q_i, s_i, q'_i, s'_i \rangle \mid i = 1, \dots, k\}$ be a transition set, $X = X(T)$ be a set of words satisfying transitions from the set T . Let $\{\langle A_i, q_i, s_i, q'_i, s'_i, \alpha_i(x) \rangle \mid i = 1, \dots, k\}$ denote the full clock characterization of a word x , for $x \in X$. Then, there exists a family R of sequences of linear expressions and inequalities of c variables (c and the number of elements of R are bounded by a number which depends on k and the number of states of the automata only) such that:*

1. *Every sequence $z \in R$ consists of k expressions $z_1, \dots, z_k$ and c inequalities $n_1, \dots, n_c$ of variables $v_1, \dots, v_c$. Moreover, each inequality has the form $v_i \geq c_i$, for $c_i \in \{0, 1\}$.*
2. *For every $x \in X$, there is a sequence of linear expressions and inequalities $z \in R$, $z = \langle z_1, \dots, z_k; n_1, \dots, n_c \rangle$ and numbers $d_1, \dots, d_c \in \mathbb{N}$ such that $\alpha_i(x) = z_i(d_1, \dots, d_c)$ for $i = 1, \dots, k$ and $n_i(d_1, \dots, d_c)$ is satisfied, for $1 \leq i \leq c$.*
3. *For every $d_1, \dots, d_c \in \mathbb{N}$ and each $z = \langle z_1, \dots, z_k; n_1, \dots, n_c \rangle \in R$, if $n_i(d_1, \dots, d_c)$ holds for $1 \leq i \leq c$, then there is a word $x \in X$ such that $\alpha_i(x) = z_i(d_1, \dots, d_c)$, for $1 \leq i \leq k$.*

Proof. We define an auxiliary one-way finite automaton A with the set of states Q^k and initial state $(q_0^1, \dots, q_0^k)$. It works as follows. If A is in a state $\langle q_1, \dots, q_k \rangle$ on the symbol a which is not an endmarker, then in one step it moves right and reaches a state $\langle q'_1, \dots, q'_k \rangle$ such that for $i \in \{1, \dots, k\}$, if the automaton A_i starts the computation in the state q_i on the symbol a , then it enters the next cell in state q'_i . Every transition of A given by $\delta(\langle q_1, \dots, q_k \rangle, a) = (\langle q'_1, \dots, q'_k \rangle, 1)$ is associated with a *step vector* $(s_1, \dots, s_k)$ such that for $i \leq k$ automaton A_i needs s_i steps in order to move right from the state q_i while reading the symbol a . Now, we define a directed multi-graph G representing the transition function of A and its steps vectors. The vertices of the graph represent the states of A ; the edges represent transitions of A . Each edge is labeled by an input symbol a and a step vector $(s_1, \dots, s_k)$ associated to this transition. We identify an edge by a tuple (v_1, v_2, a) , where v_1, v_2 are vertices adjacent to the edge (v_1 is the source) and a is a symbol labeling the edge. For an edge (v_1, v_2, a) , for $i \in \{1, \dots, k\}$, let $s_i(v_1, v_2, a)$ denote the i th coefficient of the *steps vector* of this edge.

Computation of A on input $x = x(1)x(2) \dots x(p)$ (hence, also of $A_1, \dots, A_k$) is represented by a path $W = (w_1, w_2, x(1))(w_2, w_3, x(2)) \dots (w_p, w_{p+1}, x(p))$ in G which starts and finishes in vertices representing appropriate states, say vertices s and t . Then, A_i makes $\sum_{j=1}^p s_i(w_j, w_{j+1}, x(j))$ steps on x . We denote this number by $s_i(W)$ or $s_i(x)$.

Our goal is to construct a family R of sequences of expressions and inequalities over some fixed set of variables V which “linearly characterize” all paths from s to t . Moreover, the absolute values of coefficients in expressions and inequalities as well as the number of elements of R are to be bounded by numbers which depend only on the size of G . It means that for every path W from s to t there is a sequence $\langle z_1, \dots, z_k, n_1, \dots, n_c \rangle \in R$ and values $\mathcal{V}$ for variables V such that inequalities $n_1, \dots, n_c$ are satisfied for $\mathcal{V}$ and $s_i(W) = z_i(\mathcal{V})$ for each i .

The proof is by induction on the number of vertices in G . We partition each path into subpaths going into the graph $G' = G \setminus \{s\}$. Then, using induction hypothesis we combine “linear descriptions” of paths in G' into a description of paths from s to t in G .

The claim is obviously true for graphs with one vertex. Now, assume that it holds for all graphs with less than n vertices. Let G be a graph with n vertices, $U = (u_1, u_2, \delta_1) \dots (u_{p-1}, u_p, \delta_{p-1})$ be a path in G , $u_1 = s$ and $u_p = t$. Let $U = U_1 U_2 \dots U_a$ be the coarsest partition of U such that s may occur only on the first and the last edge of any U_i . So for every i , the path U_i without the last and the first edge does not leave the graph $G' = G \setminus \{s\}$. In this way we partition any path U from s to t into subpaths going through the graph G' with a smaller number of vertices and some edges “connecting” these subpaths. By the induction hypothesis, these subpaths have appropriate “linear characterizations”. Our strategy is to use them for constructing such a “linear characterization” for paths from s to t in G . Obviously, the linear characterization should have the size bounded by some value dependent on the size of the graph, independent of the lengths of paths. However, the partition of U may depend on its length. In order to overcome this problem, we group subpaths in categories. A category is defined by:

- (a) the first and the last edge: (s, u_1, σ_1) and (u_2, v, σ_2) (where $v \in \{s, t\}$) of the subpaths of this category,
 - (b) a sequence of linear expressions and inequalities $r = \langle w_1, \dots, w_k; n_1, \dots, n_m \rangle$ for paths from u_1 to u_2 in G' (such a sequence exists by the induction hypothesis).
- Observe that, by the induction hypothesis, the number of categories is limited by a number which depends solely on the size of the graph (so, on the number of states of automata). Let $cat(U_i)$ denote the category of the subpath U_i , $K(U)$ denote the set of all categories of subpaths $U_1, \dots, U_a$.

Now, we show how to construct a sequence $\langle w_1, \dots, w_k; n_1, \dots, n_m \rangle$ of equalities and inequalities which characterizes all paths U' from s to t in G such that $K(U') = K(U)$. So, by considering all possible sets of categories we obtain a family of sequences which linearly characterize all possible paths from s to t in G .

Let $K(U) = \{K_1, \dots, K_p\}$. We take one category, say K_l , and construct the sequence of expressions and inequalities which will “cover” all subpaths of this category in U and similarly all subpaths of this category in every path U' from s to t such that $K(U') = K(U)$. Assume that the category K_l is characterized by:

- (a) edges (s, u_1, σ_1) and (u_2, s, σ_2) (we omit the case in which the subpath terminates in t ; it can be considered in a similar way),
- (b) linear expressions $w_i = a_{i0} + \sum_{j=1}^m a_{ij} \xi_j$ for $i \in \{1, \dots, k\}$, and inequalities $n_i \equiv (\xi_i \geq c_i)$, for $i \in \{1, \dots, d\}$ where $c_i \in \{0, 1\}$, over variables $\xi_1, \dots, \xi_m$.

Let $B = \{i \mid c_i = 1\}$. Our goal now is to construct expressions and inequalities such that all steps of the automata made “inside” subpaths of category K_l will be “counted” by these expressions. The key point is, we do not care where these subpaths occur, and how they exactly look like. We only count the total number of steps inside them. We claim that inequalities $n'_0 \equiv (\xi_0 \geq 1)$, $n'_i \equiv (\xi_i \geq 0)$ for $i \in \{1, \dots, m\}$, and expressions

$$w'_i = \left(a_{i0} + \sum_{j \in B} a_{ij} + s_i(s, u_1, \sigma_1) + s_i(u_2, s, \sigma_2) \right) \xi_0 + \sum_{j=1}^m a_{ij} \xi_j$$

over variables $\xi_0, \dots, \xi_m$ may be used for this purpose. Intuitively, the “new” variable ξ_0 denotes the number of subpaths of category K_l in the path; elements summarized as a coefficient multiplied by ξ_0 consist of all “elements” which have to occur in every (sub)path of category K_l . The remaining variables are responsible for the “optional” parts of the paths (possibly repeated many times).

Finally, we join expressions and inequalities of all categories $K_1, \dots, K_p$.

Let N_i denote the sequence of inequalities and $W_i = \langle w_i^1, \dots, w_i^k \rangle$ be the sequence of expressions which characterize the subpaths of category K_i (to avoid conflicts, each category has its own variables). The sequence of inequalities and expressions which characterizes the set of paths U' from s to t such that $K(U') = K(U)$ contains the inequalities from N_i , over all i , and the expressions $\sum_i w_i^1, \dots, \sum_i w_i^k$. It follows immediately from the construction that this sequence has all properties claimed. ■ Lemma 2

Now, in order to complete the proof of Linear Characterization Lemma we show how to construct a system of linear equations and inequalities which defines the number of steps spent on silent blocks and dependencies between them.

Let $\langle i, (q_1, j_1), (q_2, j_2), \dots, (q_l, j_l) \rangle$ and $\langle i+1, (q'_1, j'_1), (q'_2, j'_2), \dots, (q'_l, j'_l) \rangle$ be consecutive entries in the communication pattern H for any $i \in \{1, \dots, g-1\}$. Let A_α be the automaton which sends the message in the $(i+1)$ st communication configuration (if more than one automaton sends a message we pick any of them). Let $I \subseteq \{1, \dots, l\}$ be the indices of automata which are then in blocking states. Then, the properties

$$t_{\alpha, j_{\alpha+1}}(x) + t_{\alpha, j_{\alpha+2}}(x) + \dots + t_{\alpha, j'_\alpha}(x) \geq t_{\delta, j_{\delta+1}}(x) + t_{\delta, j_{\delta+2}}(x) + \dots + t_{\delta, j'_\delta}(x) \text{ for } \delta \in I,$$

$$t_{\alpha, j_{\alpha+1}}(x) + t_{\alpha, j_{\alpha+2}}(x) + \dots + t_{\alpha, j'_\alpha}(x) = t_{\delta, j_{\delta+1}}(x) + t_{\delta, j_{\delta+2}}(x) + \dots + t_{\delta, j'_\delta}(x) \text{ for } \delta \notin I$$

determine behaviour of system S between the i th and the $(i+1)$ st communication configuration. By Lemma 2, the variables t_{ij} in the above expressions may be replaced by some linear expressions of a constant number of variables (attached with some inequalities concerning these variables). This yields the inequalities stated by Lemma 2. It is easy to see that they satisfy all properties claimed.

3.3 Outline of diophantine characterization of two-way automata

In this section, we confine ourselves to these elements that differ from the construction for the one-way case. Before we go into details, we sketch how to recognize language L_2 , which witnesses that $O(\log \log n)$ messages may suffice in the case of two-way automata. Simultaneously, it explains in which direction the changes in the construction of diophantine systems must go. We use automata A_1 and A_2 to recognize L_2 . First, A_1 checks if the first two blocks of ones have lengths 2 and 3. Then, for $i = 3, 4, \dots$ they verify that all required divisibilities hold. In order to check whether $f_{i-1} \mid (f_i - 1)$ (and similarly if $f_{i-2} \mid (f_i - 1)$), automata A_1 and A_2 start moving with the same speed through block $i-1$ and i . Automaton A_1 goes from the left to the right in block i , automaton A_2 loops between the ends of block $(i-1)$. They terminate this stage of computation when A_1 reaches symbol $\#$ and sends a message. Since A_2 may remember in states, if it has left the symbol $\#$ in the previous step, the divisibility can be easily checked.

Now, we move to the general diophantine construction. Consider the number of steps needed to move through a silent block of input x . The first problem is that each automaton may move through a silent block many times entering it from left/right and leaving on the left/right side. However, the number of possible transitions over a silent block is limited by the number of possible “starting points” determined by the state of the automaton and the side on which it enters the block. (A head cannot loop inside a silent block, since finally a message would be sent when the head is inside the block.) So, we have variables $t_{i,j,q,d}$ denoting the number of steps which the automaton A_i spends before leaving the j th silent block, when entering it in the state q from the side d .

In order to describe dependencies between the variables $t_{i,j,q,d}$, we have to use divisibilities. This is due to the fact that, as in the example above, while the automaton which sends the i th message moves through some blocks, the other automata may loop over some other blocks. For example, assume that the i th message is sent by automaton A_α , which during the part of computation between the $(i-1)$ st and the i th message makes T_α steps (T_α is obviously a sum of some values $t_{\alpha,*,*,*}$). At this time, automaton A_β , after some initial computation consisting of $T_{\beta,1}$ steps gets into a loop consisting of $T_{\beta,2} > 1$ steps (by loop we mean a part of computation which starts and finishes in state q on position p for any q and p). Obviously, one may define beginning of the loop to get that A_β is at the end of the loop when a message is sent by A_α . $T_{\beta,1}$ and $T_{\beta,2}$ are obviously sums of some values $t_{\beta,*,*,*}$. The dependencies described take the form $T_{\beta,2} | (T_\alpha - T_{\beta,1})$. In the case that A_β does not get into a loop, we express the dependencies like for systems of one-way automata.

The next major change with respect to the one-way case is how to construct linear formulas for the variables $t_{i,j,q,d}$. Let us consider a transition $\langle A_i, q, d, q', d' \rangle$ of block j . First, we construct a (nondeterministic) one-way automaton for such a transition. This automaton goes from left to right and guesses crossing sequences for consecutive positions (similarly as it is done to show equivalence between one-way and two-way finite automata [6]; crossing sequences required for border positions of the block are determined by the tuple $\langle A_i, q, d, q', d' \rangle$). With each step of this automaton we associate a weight denoting the number of steps needed by A_i to perform transitions described by the crossing sequence. The automaton is nondeterministic, but its work corresponds to a path in a certain graph. The rest of the construction is the same as before (so, we built a “Cartesian product” of such automata associated to transitions).

4 Solutions for diophantine problems for addition and divisibility

Decidability issues. In general, deciding whether a system of diophantine inequalities has a solution is an undecidable problem [11]. Even more surprising is that very “simple” systems of diophantine inequalities are undecidable. There have been lot of research on setting the exact boundary between decidable and undecidable instances of this problem. Decidability proofs follow often a common scenario: one shows that if there is a solution of a system of diophantine inequalities, then there is a solution bounded by some number that can be effectively determined. So deciding whether there is a solution may be reduced to checking whether there is a solution of a certain size.

Minimal solutions for polynomials of degree 1. In this case, the system consists of a number of linear equations and inequalities.

Theorem 3. ([4], a restricted version) *Let A, b, C, d be respectively $m \times n$, $m \times 1$, $p \times n$, $p \times 1$ matrices with integer coefficients with absolute values bounded by a constant f . If there exists an integer solution x for $Ax = b$ and $Cx \geq d$, then there is a solution x' with absolute values of coefficients bounded by $2^{c_f f^n}$, where c_f is a constant depending on f .*

Minimal solutions for divisibility. Now consider a system ϕ of divisibility relations

$$f_i(x) | g_i(x), \quad \text{for } i \leq K, \quad (1)$$

where each f_i and g_i is a polynomial of degree 1 and variables from x are over $\mathbb{N}$. Recall that an equality $f(x) = g(x)$ may be expressed as divisibilities: $f(x) | g(x)$ and $g(x) | f(x)$, $f(x) + 1 | g(x) + 1$, $g(x) + 1 | f(x) + 1$. An inequality $f(x) > g(x)$ may be expressed by an equality $f(x) + y + 1 = g(x)$ with a new variable y , and then in turn by divisibilities.

In papers [9, 10], Lipshitz discusses the problem of existence of natural solutions of systems (1). In [9], he shows that the problem is decidable by reducing it to the question of existence of solutions modulo p^s for certain prime numbers p and $s \in \mathbb{N}$ which depends on the system. In [10], he argues that for each fixed K , the minimal solutions, if exist, are of a polynomial size. Note that from the example of language L_2 it follows that the degree of this polynomial grows with K .

For our purposes, we follow the arguments from [9] and state some properties implicitly contained there. The first step in Lipshitz's construction is to replace a system of diophantine divisibilities by systems of totally positive diophantine divisibilities (for a technical definition see [9]). This special form of diophantine divisibilities is crucial for finding “small” solutions.

Theorem 4. ([9], Section 1) *For each system of diophantine divisibilities ϕ there is a set $\phi_1, \dots, \phi_m$ of totally positive systems of diophantine divisibilities, so that the set of solutions to ϕ coincides with the union of the sets of solutions for $\phi_1, \dots, \phi_m$.*

In the following parts we shall assume that ϕ is a system of diophantine divisibilities with all coefficients bounded by some constant. Let N be an upper bound on the number of variables and on the number of divisibilities occurring in ϕ . By following the original proof, we get an estimation on the size of coefficients:

Proposition 1. *There is a polynomial w such that absolute value of each coefficient occurring in any ϕ_i is bounded by $2^{2^{w(N)}}$.*

The next step in the construction of Lipshitz is to find a “small” solution for ϕ_i provided that we know that a solution does exist. This is given by the next two theorems:

Theorem 5. ([9], Lemma 5) *If ψ is a system of totally positive diophantine divisibilities, p is a prime number, and there is a solution to ψ , then there is a solution $\bmod p^{k_p}$, such that for every divisor $f_i(x)$ from ψ is different from $0 \bmod p^{k_p}$. The number k_p can be bounded in terms of the maximum value of coefficients in ψ , the number of divisibilities and the number of variables.*

Proposition 2. *If we start with a ϕ as above, then p^{k_p} is bounded by $2^{2^{v(N)}}$ for some polynomial v .*

Theorem 6. ([9], **Lemma 6**) *Let ψ be a system of totally positive diophantine divisibilities. There is $M_\phi \in \mathbb{N}$ depending on the maximum value of coefficients, the number of divisibilities and the number of variables in ψ such that the following property holds. If for every prime number $p < M_\phi$, there is a solution of ψ modulo p^{k_p} such that every divisor $f_i(x)$ in ψ holds $f_i(x) \not\equiv 0 \pmod{p^{k_p}}$, then there is an integer solution to ψ .*

Again, following the construction we may check the following property:

Proposition 3. *The integer solution found in Theorem 6 for the system ϕ is bounded by $2^{2^{u(N)}}$, where u is a polynomial.*

Conclusions Still, there is a gap between the bound $\Theta(\log \log n)$ given by L_2 and the lower bound of Theorem 2. To close it, one may try to sharpen the results concerning minimal solutions of systems of diophantine divisibilities (compare also [10] and [2]).

Acknowledgment. We would like thank Krzysztof Loryś for very helpful discussions.

References

1. P. Beame, M. Tompa, P. Yan, *Communication-Space Tradeoffs for Unrestricted Protocols*, SICOMP 23 (1994), 652–661.
2. P. J. Cohen, *Decision procedures for real and p -adic fields*, Comm. on Pure and Applied Math. 22 (1969), 131–151.
3. E. Contejean, H. Devie, *An efficient algorithm for solving systems of diophantine equations*, Information and Computation 113 (1994), 143–172.
4. J. von zur Gathen, M. Sieveking, *A bound on solutions of linear integer equalities and inequalities*, Proc. of the AMS 72(1) (1978), 155–158.
5. M. Holzer, *Multi-Head Finite Automata: Data-Independent Versus Data-Dependent Computations*, Proc. MFCS'97, LNCS 1295, Springer Verlag, Berlin, 1997, 299–309.
6. J. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.
7. T. Jurdziński, *Communication Aspects of Computation of Systems of Finite Automata*, Wrocław University, 2000. (<http://www.ii.uni.wroc.pl/~acm/doktoraty.html>)
8. T. Jurdziński, M. Kutylowski, K. Loryś, *Multiparty finite computations*, in Computing and Combinatorics, Proc. COCOON'99, LNCS 1627, Springer Verlag, Berlin, 1999, 318–329.
9. L. Lipshitz, *The diophantine problem for addition and divisibility*, J. AMS 235 (1978), 271–283.
10. L. Lipshitz, *Some remarks on the diophantine problem for addition and divisibility*, Bull. Soc. Math Belg. 33 (1981), 41–52.
11. Ju. Matijasevič, *Hilbert's tenth problem*, Foundations in Computing Series, MIT Press, Cambridge, 1993.
12. V. Mitrana, *On the degree of communication in parallel communicating finite automata systems*, Journal of Automata, Languages and Computation Vol. 5, 3(2000), 301–314.