RWTH Aachen

Department of Computer Science
Technical Report

3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata

Henry N. Adorna

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2002-06

RWTH Aachen · Department of Computer Science · February

The publications of the Department of Computer Science of RWTH Aachen (*Aachen University of Technology*) are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata *

Henry N. Adorna

Lehrstuhl für Informatik I
RWTH Aachen, Germany
and
Department of Mathematics, Univ. of the Philippines,
Diliman 1101 Q.C., Philippines
Email: henri@{i1.informatik.rwth-aachen.de, math.upd.edu.ph}

Abstract. Despite the facts that automata theory is one of the oldest and most extensively investigated areas of theoretical computer science, and finite automaton is the simplest model of computation, there are still principal open problems about finite automata. One of them is to estimate, for a regular language L, the size of the minimal nondeterministic finite automaton accepting L. Currently, we do not have any method that would at least assure an approximation of this value. The best known technique for proving lower bound on the size of the minimal nondeterministic finite automata is based on communication and this technique covers all previously used approaches. Unfortunately, there exist regular languages with an exponential gap between the communication complexity lower bound and the size of a minimal nondeterministic finite automaton. The contribution of this paper is to improve the communication complexity lower bound technique in order to get essentially better lower bounds for some regular languages.
Keywords: Finite Automata, Communication Complexity, Uniform Protocol, Lower Bound

1 Introduction

Automata theory is one of the oldest and most extensively investigated area of theoretical computer science and finite automaton is the simplest model of computation. Old as it is, there are still principal open problems about it, e. g., for a regular language L, we would like to estimate the size of the minimal nondeterministic finite automaton (NFA) accepting L. At the moment, we do not have any method that would at least assure an approximation of this value. This is in contrary to the known result for the deterministic case, where the minimum deterministic finite automaton (DFA) for L can be determined in time $O(n\,log\,n)$ [9]. The problem of determining, for a regular language L, the size of the minimal NFA accepting L is PSPACE-complete [6] except when the alphabet is a singleton set. The minimal NFA is not necessarily unique [3, 16]. Procedures

* Supported by Deutscher Akademischer Austauschdienst(DAAD) and DFG Project HR 14/16-1

for this minimization problem are investigated in [19] and in [22] a heuristic for this search was introduced and implemented in AMoRE[1]. The search for the general procedure is not likely to be successful, since the problem of converting a DFA for a regular language L with k states to an NFA, also accepting L with at most k number of states is PSPACE-complete [17], too.

In the series of papers, Shallit et al. [8, 25–27] used the concept of automaticity in measuring the number of states of the minimal NFA for an n^{th} order approximation to a regular language $L \subseteq \Sigma^*$, denoted by $N_L(n)$. They proved that, for an alphabet with at least two letters, $N_L(n) = O(k^{\frac{n}{2}})$, where k is the cardinality of the alphabet and n is the (maximum) length of the string, also that for almost all languages L and every $\epsilon > 0$, $N_L(n) > \frac{(1-\epsilon)k^{\frac{n}{2}}}{\sqrt{k-1}}$, for sufficiently large n [27]. In the case, when the alphabet is a singleton set, their result says that, there exists a constant c for almost all unary language L, such that $N_L(n) \geq \frac{cn}{log\,n}$, for all sufficiently large n, where n is the same as above [25]. In the third of the series, they used the idea of communication complexity in giving lower bound on the minimal NFA. Unfortunately, the obtained lower bound is not always tight [7, 8]. In this work, we also base our investigation on communication complexity, but we consider uniform protocols.

The communication complexity of 2-party protocols was introduced by Yao [30] (also implicitly considered by Abelson [1]) in the late 70's. The initial goal was to develop a method for proving lower bounds on the complexity of distributive and parallel computations [15, 20, 28, 29]. Due to the development of nontrivial mathematical machinery for determining the communication complexity of concrete problems [2, 4, 11, 15, 20, 21, 23, 24], it has established itself as a sub-area of complexity theory. The primary contributions of the study of communication complexity lie particularly in proving lower bounds on the complexity of solving specific problems on several fundamental computing models, and in comparing the power of different modes of computation.

The first simple application of communication complexity for proving lower bounds on the size of the minimal deterministic, resp. nondeterministic finite automaton, $\mathrm{s}(L)$ resp. $\mathrm{ns}(L)$ was shown in [12], i.e. $\mathrm{s}(L) \geq 2^{\mathrm{cc}_1(L)}$ resp. $\mathrm{ns}(L) \geq 2^{\mathrm{ncc}_1(L)}$, for every regular language $L \subseteq \Sigma^*$, where $\mathrm{cc}_1(L)$ resp. $\mathrm{ncc}_1(L)$ are the 1-way deterministic resp. nondeterministic communication complexity of L (see [12, 15] for details). The technique mentioned in [7] is a special case of these communication complexity method. But because of the non-uniformity of the protocol it cannot completely captures the complexity of the uniform acceptance of L. Hromkovič, et al. [5, 15, 13] extended the non-uniform model of communication protocols for computing finite functions to a uniform model recognizing languages. This gave a closer relationship between communication complexity and finite automata and resulted in successful application of several results about communication complexity to uniform computing models like automata.

[1] Available via anonymous ftp at: `ftp.informatik.uni-kiel.de:pub/kiel/amore`

In [5, 11] the authors considered the amount of messages that a 2-party protocol will need to recognize a language and they call it the **message complexity of that protocol**, denoted $\mathrm{mc}(P_2)$. The message complexity of the best 2-party protocol is called the **message complexity of the language** L, denoted $\mathrm{mc}(L)$. They showed that the message complexity of a 1-way uniform 2-party deterministic protocol for L, denoted 2-dmc(L) is exactly the size of the minimal deterministic finite automaton for L, i. e., 2-dmc$(L) = \mathrm{s}(L)$. While 2-nmc(L), which is the message complexity of the best 1-way uniform 2-party nondeterministic protocol for L is the lower bound for $\mathrm{ns}(L)$, still. Until now the best known method for proving lower bounds on the size of minimal NFA's is based on uniform nondeterministic communication complexity [11, 15] and all other known methods are special cases of this method. For some overview on this topic see also [10].

Recently, Hromkovič, et al. [14] showed that there exist a sequence of languages such that its 2-party nondeterministic message complexity is weak in providing lower bound for its minimal nondeterministic finite automaton. Their results showed that for a sequence $\{L_{3n}\}_{n=1}^{\infty}$ of regular languages the 2-party nondeterministic message complexity of L_{3n} is much smaller than the size of the minimal nondeterministic finite automata for L_{3n}. The same result was obtained by Jirásková [18] using another regular language.

In this paper we try to improve the ability of message complexity to provide lower bound by introducing the natural extension of 1-way uniform 2-party protocols, i.e.,**1-way uniform k-party protocols**, as a new model of computation. A 1-way uniform k-party protocol is a uniform (deterministic or nondeterministic) protocol between k linearly ordered computers that communicate their messages among neighbors in a left-to-right (only) mode via a communication link and the last computer has to decide about acceptance of the input. We define k-dmc(P_k) resp. k-nmc(P_k) as the **k-party deterministic**, resp. **nondeterministic message complexity of a protocol** P_k. These are the total amount of messages that is required of P_k to recognize a language. And the message complexity of the best deterministic resp. nondeterministic protocol P_k recognizing L is the **k-party deterministic** resp. **nondeterministic message complexity of** L, denoted k-dmc(L) resp. k-nmc(L).

Our result shows that in the deterministic case, the k-dmc(L) is equal to $\mathrm{s}(L)$, too. We show that in the nondeterministic case k-nmc(L) also provides a lower bound for the size of the minimal NFA for L. Our main contribution is in showing that 3-nmc(L) can provide essentially better lower bounds for $\mathrm{ns}(L)$ than 2-nmc(L) for some languages. In particular, we showed that there is a sequence of regular languages $\{L_{3n}\}_{n=1}^{\infty}$, such that 3-nmc$(L_{3n}) = 2^{\Omega(\sqrt{(2\text{-nmc}(L_{3n}))}} \leq \mathrm{ns}(L_{3n})$.

This paper is organized as follows. In Section 2, we give the definitions of the classical model of a one-way finite automata, both DFA and NFA, and the 2-party deterministic and nondeterministic protocols. We end the Section by stating some relationships between automata and communication complexities. We introduce the k-party protocols, and its message complexities in Section 3 and give the relation between the k-party protocols and finite automata. The next section shows that 3-party nondeterministic message complexity can be better than the 2-party ones as lower bound for $ns(L_{3n})$. Finally, we give few remarks on k-nmc(L).

2 Preliminaries

In this paper, we consider the standard one-way models of deterministic finite automata(DFA's) and nondeterministic finite automata (NFA's).

2.1 Finite Automata and Communication Complexity

Definition 1 *A* **deterministic finite automaton (DFA)** *M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where*

- *Q is a finite set of states,*
- *Σ is the set of alphabet,*
- *$F \subseteq Q$, is the finite set of final or accepting states,*
- *$q_0 \in Q$ is the initial state of M,*
- *δ is transition function from $Q \times \Sigma \longrightarrow Q$.*

We say that the input $x \in \Sigma^*$ is **accepted by** *M if $\hat{\delta}_M(q_0, x) \in F$ where $\hat{\delta}_M : Q \times \Sigma^* \longrightarrow Q$, such that*

- *$\hat{\delta}_M(q, \lambda) = q$ for every $q \in Q$, and,*
- *$\hat{\delta}_M(q, wa) = \delta(\hat{\delta}(q, w), a)$ for every $w \in \Sigma^*$ and every $a \in \Sigma$.*

The **language accepted by** *M is the set $L(M) = \{x \in \Sigma^* | \hat{\delta}(q_0, x) \in F\}$.*
The **size of** *M is the number of states in M, denoted by $\mathrm{s}(M) = |Q|$.*

Definition 2 *A* **nondeterministic finite automaton (NFA)** *is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where each of Q, Σ, F and q_0 are defined similarly as above, while δ is a* **transition relation** *from $Q \times \Sigma \longrightarrow 2^Q$, where 2^Q is the power set of Q.*

An input word $x \in \Sigma^*$ is said to be **accepted by** *A, whenever $\hat{\delta}_A(q_0, x) \cap F \neq \emptyset$ where $\hat{\delta}_A : Q \times \Sigma^* \longrightarrow 2^Q$ such that*

- *$\hat{\delta}_A(q, \lambda) = \{q\}$ for every $q \in Q$, and*
- *$\hat{\delta}_A(q, wa) = \{p \in Q | p \in \delta(r, a)$ for some $r \in \hat{\delta}_A(q, w)\}$ for every $q \in Q$ and every $a \in \Sigma$, $w \in \Sigma^*$.*

The **language accepted by** *A is the set $L(A) = \{x \in \Sigma^* | \hat{\delta}_A(q_0, x) \cap F \neq \emptyset\}$*
The **size of** *A is denoted by $\mathrm{s}(A) = |Q|$.*

Definition 3 *A language L is said to be* **regular** *iff there is a finite automata (FA) M that accepts it, i.e. $L = L(M)$. For every regular language L, we define*

$$\mathrm{s}(L) = min\ \{\mathrm{s}(M) | M \text{ is a DFA such that } L(M) = L\},$$

and

$$\mathrm{ns}(L) = min\ \{\mathrm{s}(A) | A \text{ is an NFA such that } L(A) = L\}.$$

An fa B is said to be **minimal for** *L if $L(B) = L$ and $\mathrm{s}(L) = \mathrm{s}(B)$.*

The first simple application of communication complexity for proving lower bounds on the size of the minimal deterministic, resp. nondeterministic finite automaton, $\mathrm{s}(L)$ resp. $\mathrm{ns}(L)$ was shown in [12], i.e. $\mathrm{s}(L) \geq 2^{\mathrm{cc}_1(L)}$ resp. $\mathrm{ns}(L) \geq 2^{\mathrm{ncc}_1(L)}$, for every regular language $L \subseteq \Sigma^*$, where $\mathrm{cc}_1(L)$ resp. $\mathrm{ncc}_1(L)$ are the 1-way deterministic resp. nondeterministic communication complexity of L (see [12, 15] for details). The technique mentioned in [7] is a special case of these communication complexity method. But because of the non-uniformity of the protocol it cannot completely captures the complexity of the uniform acceptance of L.

2.2 1-Way Uniform 2-Party Protocols

With the aim of trying to relate closely the idea of automata and communication complexity Hromkovič [15] and Duriš et al. [5] introduced the concept of uniformity in the 2-party communication protocol model for recognizing languages.

We observed that the main difference between uniform protocols and (standard) protocols is the way the input is partitioned between the participating computers, say C_1 and C_2, in the case of 2-party model. Suppose a protocol P_2 computes a Boolean function $f : \{0,1\}^r \times \{0,1\}^s \longrightarrow \{0,1\}$. It can be viewed as the partition of inputs of f from $\{0,1\}^{r+s}$ into the prefix of r bits and a suffix of s bits (i.e., assigning the first r bits to C_1 and the rests to C_2) and a communication between C_1 and C_2 in order to compute the value of f. A uniform protocol over Σ considers, for every input $x = x_1x_2x_3 \ldots x_n \in \Sigma^n$, $n+1$ partitions of x, namely $(\lambda, x)(x_1, x_2x_3 \ldots x_n), (x_1x_2, x_3 \ldots x_n), \ldots, (x_1x_2 \ldots x_{n-1}, x_n), (x, \lambda)$. For each of these partitions the uniform protocol must accepts or reject if $x \in L$ or not (i.e., a uniform protocol P_2 for L must recognize the membership of x to L for every partition of x between C_1 and C_2).

We repeat here the definitions given in [11, 15].

Definition 4 *Let Σ be an alphabet and $L \subseteq \Sigma^*$. A* **1-way uniform 2-party deterministic protocol** *over Σ is a pair $P_2 = \langle \Phi, \varphi \rangle$ where*

- $\Phi : \Sigma^* \longrightarrow \{0,1\}^*$ *is a function with the prefix freeness property and*
- $\varphi : \Sigma^* \times \{0,1\}^* \longrightarrow \{accept, reject\}$ *is a function.*

We say that P_2 accepts the regular language L if for all x, $y \in \Sigma^*$, $\varphi(y, \Phi(x)) = accept$ iff $xy \in L$.

The **1-way 2-party message complexity of the protocol** *P_2 is given by:*

$$\textbf{2-dmc}(P_2) = |\{\Phi(x) | x \in \Sigma^*\}|.$$

For a regular language $L \subseteq \Sigma^*$, the **1-way 2-party deterministic message complexity of** *L is given by:*

$$\textbf{2-dmc(L)} = min\ \{2\text{-dmc}(P_2)\ |P_2 \textit{ is a 1-way uniform 2-party deterministic protocol accepting } L\}$$

We give the following fact, which was shown in [5, 11].

Lemma 1. [5, 11] *For every regular language L, 2-dmc(L)=s(L.)*

Definition 5 *Let Σ be an alphabet and $L \subseteq \Sigma^*$. A* **1-way uniform 2-party nondeterministic protocol** *over Σ is a pair $P_2 = \langle \Phi, \varphi \rangle$ where*

- $\Phi : \Sigma^* \longrightarrow 2^{\{0,1\}^*}$, *where*
  - *$\Phi(x)$ is a function with the prefix freeness property.*
  - *For all $x \in \Sigma^*$, $\Phi(x)$ is a finite set.*
  - *The set $\{\Phi(x) | x \in \Sigma^*\}$ is finite, and*
- $\varphi : \Sigma^* \times \{0, 1\}^* \longrightarrow \{accept, reject\}$ *is a function.*

The protocol P_2 accepts the regular language L if for all x, $y \in \Sigma^*$, $\exists\, z \in \Phi(x)$ such that $\varphi(y, z) = accept$ iff $xy \in L$.

The **1-way 2-party (nondeterministic) message complexity of the protocol** *P_2 is given by:*

$$\textbf{2-nmc}(P_2) = |\{\Phi(x) | x \in \Sigma^*\}|.$$

For a regular language $L \subseteq \Sigma^*$, the **1-way 2-party nondeterministic message complexity of** *L is given by:*

$$\textbf{2-nmc(L)} = min\ \{2\text{-nmc}(P_2)\ |P_2 \textit{ is a 1-way uniform 2-party nondeterministic protocol for } L\}$$

The following fact is a straightforward extension of the relation among protocols and automata from [15].

Lemma 2. [15] *For every regular language L, 2-nmc(L) $\leq$ ns(L.)*

Glaister and Shallit [7] also proved a theorem which provides a lower bound based on communication complexity, but it is a special case of Remark 2, even a special case of the approach based on [12].

Another important relation between automata and protocols was established in [14], where it was shown that two-way deterministic communication complexity can be used for proving lower bounds on the size of nondeterministic finite automata with bounded unambiguity.

3 k-Party Protocols and Finite Automata

The natural extension of this model is to allow the partition of the input into k pieces assigned to k computers. The first computer passes its message to the second computer, then it (the second computer) transmits its message to the third computer depending on its part of the input and the message received from the first computer, and so on until it reaches the point where the $(k-1)^{th}$ computer gives its message to the k^{th} computer depending on its part of the input and the message received from the $(k-2)^{th}$ computer. Finally, the k^{th} computer has to decide on the acceptance of the input. In this model, we leave the sole responsibility of deciding on the acceptance of the input to the k^{th} computer alone.

For instance, if we have an input $w \in \Sigma^*$, $w = x_1 x_2 \dots x_k$, $\forall\, x_i \in \Sigma^*$, the i^{th} computer gets the input part x_i, and the computation is executed as depicted at Figure 1.

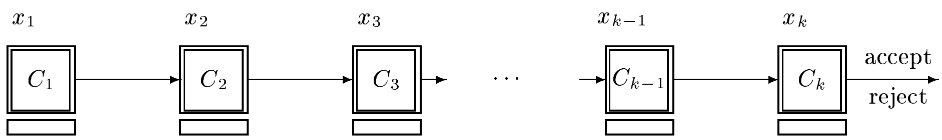

Fig. 1. The One-Way Uniform k-Party Protocol where $w = x_1 x_2 x_3 \cdots x_{k-2} x_{k-1} x_k$

Definition 6 *Let $w \in \Sigma^*$. We define a k-**partition** of w as the k-tuple of words $(x_1, x_2, x_3, \dots, x_k)$ such that $w = x_1 x_2 x_3 \dots x_k$. We allow the length of each $x_i, 1 \leq i \leq k$ to range from 0 to $|w|$.*

As an illustration of our k-partition, we consider the word $w = \alpha_1\alpha_2\alpha_3$, $\alpha_i \in \Sigma$. The following are all its 3-partitions, where λ is the empty word:

$$(\lambda, \lambda, \alpha_1\alpha_2\alpha_3), (\lambda, \alpha_1, \alpha_2\alpha_3), (\lambda, \alpha_1\alpha_2, \alpha_3), (\lambda, \alpha_1\alpha_2\alpha_3, \lambda), (\alpha_1, \alpha_2\alpha_3, \lambda),$$
$$(\alpha_1\alpha_2, \alpha_3, \lambda), (\alpha_1\alpha_2\alpha_3, \lambda, \lambda), (\alpha_1\alpha_2, \lambda, \alpha_3), (\alpha_1, \lambda, \alpha_2\alpha_3), (\alpha_1, \alpha_2, \alpha_3).$$

Our uniform protocol P_3 must correctly decide about the acceptance for each of these partitions.

We shall call this natural extension as the **1-way uniform k-party deterministic (nondeterministic) protocol** over Σ. And we give the formal definition below:

Definition 7 *For any regular language $L \subseteq \Sigma^*$, a* **1-way uniform k-party deterministic protocol** *is a triple $P_k = \langle \Phi, \{\phi_i\}_{i=2}^{k-1}, \varphi \rangle$, such that;*

- $\Phi : \Sigma^* \longrightarrow \{0,1\}^*$ *is a function having the prefix-freeness property and*
- $\phi_i : \Sigma^* \times \{0,1\}^* \longrightarrow \{0,1\}^*$, *are functions with the prefix-freeness property, for $i \in \{2, \dots, k-1\}$.*
- $\varphi : \Sigma^* \times \{0,1\}^* \longrightarrow \{accept, reject\}$ *is a function.*

The **computation** *C of P_k on an input $x = x_1 x_2 \dots x_k \in \Sigma^*$, with respect to a partition $(x_1, x_2, \dots, x_k)$ is*

$$C = c_1 \$ c_2 \$ c_3 \$ \dots \$ c_{k-1} \$ c_f$$

where

- $c_1 = \Phi(x_1)$
- $c_i = \phi_i(x_i, c_{i-1})$, *for all $i \in \{2, 3, \dots, k-1\}$,*
- $c_f = \varphi(x_k, c_{k-1})$.

If $c_f = accept$, for an input w, then we say that C is an **accepting** *computation. Otherwise, C is a* **rejecting** *computation. We say that P_k accepts a regular language L, $L(P_k) = L$ if*

- *$x \in L$ iff for all partition Π of x the computation of P_k on x with respect to Π is accepting,*
  and similarly it rejects when,
- *$x \notin L$ iff for all partition Π of x the computation of P_k on x with respect to Π is rejecting.*

The **1-way uniform k-party deterministic message complexity of** *P_k is given by:*

$$\textbf{k-dmc}(P_k) = |\{\Phi(x) | x \in \Sigma^*\} \cup \{\phi_i(y, \gamma) | y \in \Sigma^*, \gamma \in \{0,1\}^*, i = 2, \dots, k-1\}|$$

For a regular language $L \subseteq \Sigma^*$, the **1-way uniform k-party deterministic message complexity of** *L is given by:*

$$\textbf{k-dmc(L)} = min \; \{ \; k\text{-dmc}(P_k) | P_k \text{ is a 1-way uniform k-party deterministic protocol for } L \}$$

The nondeterministic counterpart of these models over Σ, can be defined as follows:

Definition 8 *For any regular language $L \subseteq \Sigma^*$, a* **1-way uniform k-party nondeterministic protocol** *is a triple $P_k = \langle \Phi, \{\phi_i\}_{i=2}^{k-1}, \varphi \rangle$, such that;*

- $\Phi : \Sigma^* \longrightarrow 2^{\{0,1\}^*}$ *is a function having prefix-freeness property and $\bigcup_{x \in \Sigma^*} \Phi(x)$ is finite.*

- $\phi_i : \Sigma^* \times 2^{\{0,1\}^*} \longrightarrow 2^{\{0,1\}^*}$ *where*
  - ϕ_i *are functions with the prefix-freeness property,*
  - $\bigcup_{x\in\Sigma^*,\ y\in\{0,1\}^*} \phi_i(x,y)$ *is finite for all* $i = 2, \ldots, k-1$,
- $\varphi : \Sigma^* \times \{0,1\}^* \longrightarrow \{accept, reject\}$ *is a function.*

A **computation** C *of* P_k *on an input* $x = x_1x_2\ldots x_k \in \Sigma^*$ *according to a partition* $(x_1, x_2, \ldots, x_k)$ *is*

$$C = c_1\$c_2\$c_3\$\ldots\$c_{k-1}\$c_f$$

if

- $c_1 \in \Phi(x_1)$
- $c_i \in \phi_i(x_i, c_{i-1})$, *for* $i = 2, 3, 4, \ldots, k-1$, *and*
- $c_f \in \varphi(x_k, c_{k-1})$.

If $c_f = accept$, *then we say that the computation is* **accepting**. *Otherwise, the computation is* **rejecting**.

We say that P_k *accepts a language* L *if for every* $x \in L$, *and every partition of* x *there exists an accepting computation of* P_k *on* x, *and for every* $x \notin L$ *and every partition* Π *of* x, *all computations of* P_k *on* x *according to* Π *are rejecting.*

The **1−way uniform k−party nondeterministic message complexity of** P_k *is given by:*

$$\textbf{k-nmc}(P_k) = |\{\Phi(x) | x \in \Sigma^*\} \cup \{\phi_i(y,\gamma) | y \in \Sigma^*, \gamma \in \{0,1\}^*, i = 2, \ldots, k-1\}|$$

For a regular language $L \subseteq \Sigma^*$, *the* **1-way uniform k-party nondeterministic message complexity of** L *is given by:*

k-nmc(L) $=$ *min* $\{$ k-nmc$(P_k) | P_k$ *is a 1-way uniform k-party nondeterministic protocol for* $L\}$

Now, we shall give the relationship between finite automata and the message complexity of our computing model. It will be shown that in the deterministic and nondeterministic case, the new models provide lower bounds on the size of minimal automata.

Theorem 1 *For every regular language* L *and for every* $k \geq 2$,

$$\text{2-dmc}(L) = k\text{-dmc}(L) = \text{s}(L).$$

Proof. We only provide the sketch of the proof. In [5, 11] it is proved that 2-dmc(L)=s(L). So, it is sufficient to prove 2-dmc(L) $\leq$ k-dmc(L) $\leq$ s(L), for every $k \geq 2$.

The inequality k-dmc(L) $\leq$ s(L) is almost obvious. Let us consider $M = (Q, \Sigma, q_0, \delta, F)$ to be a minimal finite automaton accepting L. The first computer C_1 submits the code of the state $\hat{\delta}_M(q_0, x)$ if its input is x. Every computer

C_i, $2 \le i \le k-1$, sends the code of $\hat{\delta}_M(p,y)$ to C_{i+1} if its input is y and it reached the code of the state p from C_{i-1}. P_k accepts iff it reaches the code of a state q and $\hat{\delta}_M(q,z) \in F$, where z is the input of P_k. Obviously, the number of different messages of P_k is at most s(M)= s(L).

To show 2-dmc(L) $\le$ k-dmc(L) for every k, it is sufficient to follow the following strategy. If P_k is a 1-way k-party protocol, then a 2-party protocol P_2 can simulate it if the first computer of P_2 mimics the work of the first computer of P_k and the second computer of P_2 simulates the works of the last $k-1$ computers of P_k.

Theorem 2 *For every regular language $L \subseteq \Sigma^*$ and every positive integer $k \ge 2$,*

$$k\text{-nmc}(L) \le \text{ns}(L).$$

Proof. Let $A = (Q, \{0,1\}, \delta, q_0, F)$ be an NFA accepting L, such that ns(L)=ns(A). Let $x \in \Sigma^*$. Note that $x \in L(A)$ iff $\hat{\delta}_A(q_0,x) \cap F \neq \emptyset$ and $x \notin L(A)$ iff $\hat{\delta}_A(q_0,x) \cap F = \emptyset$.

Let $n = \lceil log_2|Q| \rceil$, we define the following function to represent every $q \in Q$ in binary of length n.

$$code : Q \longrightarrow \{0,1\}^n$$

Let $q \in Q \subseteq Z_0^+$, i.e., q is labeled as elements of nonnegative integers from 0 to $|Q|-1$. We define $code(q) = BIN_n^{-1}(q)$.[2]

Suppose we have two states $q, p \in Q$. Now, if

$$\begin{aligned} code(q) = code(p) &\Longleftrightarrow BIN_n^{-1}(q) = BIN_n^{-1}(p) \\ &\Longleftrightarrow BIN(BIN_n^{-1}(q)) = BIN(BIN_n^{-1}(p)) \\ &\Longleftrightarrow q = p. \end{aligned}$$

Thus, our function *code* is injective and assures us of a representation for every $q \in Q$ the way we wanted it to be.

Now, let $P_k = \langle \Phi, \{\phi_i\}_{i=2}^{k-1}, \varphi \rangle$ be a k-party protocol, such that

- $\Phi(x) = \{code(p) | p \in \hat{\delta}_A(q_0,x)\}$ for $\forall x \in \Sigma^*$.
- $\phi_i : \Sigma^* \times \{0,1\}^n \longrightarrow \{0,1\}^n$, where for $\forall y \in \Sigma^*$ and $\forall q \in Q$,

$$\phi_i(y, code(q)) = \{code(p) | p = \hat{\delta}_A(q,y)\}$$

for $i = 2, 3, \ldots, k-1$.

[2] This is defined in [15]. Let $w = w_1 w_2 \ldots w_m, w_i \in \{0,1\}, i = 1, \ldots, m$. Then the integer binary coded by word w is

$$BIN(w) = \Sigma_{i=1}^m w_i 2^{m-i}.$$

Let k, j be nonnegative integers, $k \ge \lceil log_2 j \rceil$. Then $BIN_k^{-1}(j)$ is a word from $\{0,1\}^k$ s.t. $BIN(BIN_k^{-1}(j)) = j$.

- $\varphi : \Sigma^* \times \{0,1\}^n \longrightarrow \{accept, reject\}$, where for $\forall z \in \Sigma^*$ and $\forall q \in Q$,

$$\varphi(z, code(q)) = \begin{cases} accept & \text{if } \hat{\delta}_A(q_{x_{k-1}}, x_k) \cap F \neq \emptyset \\ reject & \text{if } \hat{\delta}_A(q_{x_{k-1}}, x_k) \cap F = \emptyset. \end{cases}$$

It is apparent that every computer C_i in P_k simulates the scanning of A from the received coded state from C_{i-1} with their input and sends their obtained state in coded form to C_{i+1}. Obviously, P_k accepts exactly those words accepted by NFA A.

The amount of messages that P_k uses to accept L is exactly $|Q|$. Hence,

$$k\text{-nmc}(L) \leq |Q| = \text{ns}(L).$$

In the next section, we will show how our model will help in improving the usefulness of message complexity in providing lower bound on the size of the minimal NFA, in particular when $k = 3$.

4 3-Party is Better than 2-Party

In [14], Hromkovič et al. showed a drawback of using 2-nmc(L) for proving lower bounds on $ns(L)$, by exhibiting a regular language L_{3n} for which there is an exponential gap between 2-nmc(L_{3n}) and the size of the minimal nondeterministic finite automata for L_{3n}. L_{3n} is defined as follows:

$$L_{3n} = \{xyz \in \Sigma^* \mid x = y \vee x \neq z, x, y, z \in \{0,1\}^n\}.$$

The same result for another language was independently obtained by Jiráskova [18]. The proof of [14] shows the existence of a one-way uniform 2-party nondeterministic protocol requiring only $O(n^2)$ messages in order to accept L_{3n}, while its corresponding minimal nondeterministic finite automata has at least $2^{\frac{n}{2}}$ number of states.

In particular, their result is:

Lemma 3. [14] *There exists a sequence of regular languages* $\{L_{3n}\}_{n=1}^{\infty}$ *such that*

$$\text{ns}(L_{3n}) \geq 2^{\Omega(\sqrt{2\text{-nmc}(L_{3n})})}.$$

In this section, we will first show that at $k = 3$, the language L_{3n} is hard for our model, i.e., if we use three computers in recognizing the words in L_{3n}, then we will need exponentially many distinct messages. This shows a large gap between 2-party nondeterministic message complexity and the 3-party one.

In order to show a large gap between the two-party nondeterministic message complexity and three party one, we consider the language introduced in [14], i.e., L_{3n}, as defined earlier. We will show that there is a partition of inputs in L_{3n}, that always require to use at least $2^{\frac{n}{2}}$ different messages in order to decide on the acceptance of the inputs.

Theorem 3 *For every positive integer n,*

$$3\text{-nmc}(L_{3n}) = 2^{\Omega(\sqrt{2\text{-nmc}(L_{3n})})}.$$

Proof. Let P_3 be a 1-way uniform 3-party nondeterministic protocol accepting L_{3n}.

Let $Set_{3n} = \{uuu | u \in \{0,1\}^n\} \subset L_{3n}$.

For all $w \in Set_{3n}$, we fix the equal-sided partition $(u_1, u_2, u_3), u_i \in \{0,1\}^n$, $i = 1, 2, 3$.

Our first aim is to show that for every two different words x and y of Set_{3n}, the sets of accepting computations of P_3 on x and y with respect to the equal-sided partitions of x and y are disjoint.

Let $uuu, vvv \in Set_{3n}$ be two different words, i.e., $u \neq v$.

Let $C_{(u,u,u)} = C_{(v,v,v)} = m_1\$m_2\$accept$ be a computation of P_3, which is an accepting computation of both uuu and vvv with respect to the equal-sided partition.

This means, for all $x \in \{u, v\}, C_1$ gives to C_2 a message $m_1 \in \Phi(x)$ and for all $y \in \{u, v\}$ the message $m_2 = \phi_2(y, m_1)$, will be sent by C_2 to C_3 afterwards.

Then C_3 decides as follows: $\varphi(z, m_2) = accept,$ for each $z \in \{u, v\}$.

Now, let us construct an accepting computation $C_{(u,v,u)}$ of P_3 on uvu. Since $m_1 \in \Phi(u)$, $C_{(u,v,u)}$ can start with m_1. We know that $m_2 \in \phi_2(v, m_1)$, then $m_1\$m_2\$$ is a possible prefix of $C_{(u,v,u)}$. Eventually, because $\varphi(u, m_2) = accept$, we have

$$m_1\$m_2\$accept,$$

which is an accepting computation of P_3 on uvu with respect to the partition (u, v, u).

But $uvu \notin Set_{3n} \cap L_{3n}$, which means $uvu \notin L_{3n}$! This is a contradiction to the fact that P_3 accepts L_{3n}. Hence, these sets of accepting computations must be disjoint.

Thus, for each of the 2^n words in Set_{3n}, our protocol P_3 must have another computation with respect to the equal-sided partitions of uuu and vvv, i.e. there exists at least 2^n different accepting computation of P_3. On the other hand, there are at most $(3\text{-nmc}(P_3))^2$ accepting computation of P_3. So,

$$(3\text{-nmc}(P_3))^2 \geq 2^n,$$

which implies,

$$3\text{-nmc}(P_3) \geq 2^{\frac{n}{2}}.$$

Now since it was shown in [14] that 2-nmc(L_{3n})=$O(n^2)$, we obtain

$$3\text{-nmc}(L_{3n}) = 2^{\Omega(\sqrt{2\text{-nmc}(L_{3n})})}.$$

This shows that the 3-nmc(L_{3n}) can provide a better lower bound for the number of states of the minimal nondeterministic finite automaton for L_{3n}.

Corollary to this lower bound result is the following:

Corollary 1 *There is an exponential gap between* 2-nmc*(L_{3n}) and* 3-nmc*(L_{3n}).*

5 Remarks on k-nmc(L)

Let $L \subseteq \Sigma^*$, be any regular language and let P_k be such 1-way uniform k-party nondeterministic protocol computing L with respect to all k-partitions Π of the input. A protocol P_{k-1} can also compute L, by simulating P_k. This time, the first $k-2$ computers in P_{k-1} will be required to perform the job of the first $k-2$ computers in P_k, while the last computer in P_{k-1} simulates the activities of the last two computers in P_k. We knew from the previous theorem above, that this leads to the following inequality:

Remark 1 *For any regular language $L \subseteq \Sigma^*$ and positive integer k, $(k-1)$-nmc$(L) \leq k$-nmc(L).*

Let $L \subseteq \Sigma^*$, be a regular language. Let $w \in L$, such that w is of finite length. Let $|w| = r+1$ for some $r \in Z^+$. Let P_{r+1} be the protocol that recognizes L with respect to all $(r+1)$ partition of the input w.

By the preceding Remark, we have for $l = 1, 2, \ldots$;

$$r\text{-nmc}(L) \leq (r+l)\text{-nmc}(L).$$

At $l = 1$, every computer C_i, for all $i = 1, 2, \ldots, r+1$, will have a bit each, with respect to a $(r+1)$ partition $(x_1, x_2, \ldots, x_r, x_{(r+1)})$, where for every $i = 1, 2, \ldots, r+1$, $|w_i| = 1$.

However, if we consider a protocol $P_{(r+1)+j}$, for all $j \geq 0$ for the language L, then for every $(r+1)+j$ possible partitions, there will be at least j computers which will have no message of their own to transmit. They can simply propagate the message they obtained to the next computer. That is, they will only have an input λ and they will be no better than a reliable relay-players in the protocol, thus contributing nothing essential in the sending of messages. This implies that at this stage the message complexity will be invariant. Therefore,

$$(r+1)\text{-}nmc(L) = ((r+1)+j)\text{-}nmc(L),$$

for all $j \geq 0$.

Take $k \geq r+1$, then we can say the following:

Remark 2 *For any regular language $L \subseteq \Sigma^*$, there is a k, such that $(k+j)$-nmc$(L) = k$-nmc(L), for all $j \geq 0$.*

The main question is whether there exist regular languages with exponential gap between $(k-1)$-nmc(L) and k-nmc(L), for $k \geq 4$.

Acknowledgment

I would like to thank Juraj Hromkovič for the idea to consider k-party uniform protocols for proving lower bounds on the size of nondeterministic finite automata and for his comments on earlier versions of this paper. I also would like to thank Imrich Vrt'o for reading the final version of this paper and finally, the referees for their comments and careful reading of this paper.

References

1. H. Abelson, Lower Bounds on Information Transfer in Distributed Computations. *Proc. 19th IEEE FOCS* (1978), 151-158.
2. A.V. Aho, J.D. Ullman, M. Yannakakis. On Notions of Information Transfer in VLSI Circuits. *Proc. 15th ACM STOC,* (1983), 133-139.
3. A. Arnold, A. Dicky, M. Nivat, A Note about Minimal Nondeterministic Automata. *Bulletin of EATCS* **47** (1992) pp. 166-169.
4. M. Dietzfelbinger, J. Hromkovič, G. Schnitger, A Comparison of Two Lower Bound Methods for Communication Complexity. *Theoretical Computer Science* **168** (1996), 39-51.
5. P. Ďuriš, J. Hromkovič, J.D.P. Rolim, G. Schnitger, Las Vegas Versus Determinism for One-Way Communication Complexity,Finite Automata, and Polynomial-time Computations. In: R. Reischuk, M. Morvan (eds.) *Proc. of Sym. on Theoretical Aspects of Comp. Science,* LNCS **1200** (1997), 117-128.
6. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* Freeman, San Francisco, 1979.
7. I. Glaister,J. Shallit, A Lower Bound Technique for the Size of Nondeterministic Finite Automata. *Information Processing Letters*, **59** (1996), 75-77.
8. I. Glaister, J. Shallit, Automaticity III: Polynomial Automaticy and Context-Free Languages. *Computational Complexity* **7** (1998), 371-387.
9. J.E. Hopcroft, An $n \log n$ Algorithm for Minimizing States in a Finite Automaton. Techn. rep. STAN-CS-71-190, Stanford University, Computer Science Department, 1971.
10. J. Hromkovič, Communication Protocols - An Exemplary Study of the Power of Randomness. In: P. Pardalos, S. Kajasekaran, J. Reif, J. Rolim (eds.) *Handbook on Randomized Computing.* Kluwer Publ. 2001.
11. J. Hromkovič, G. Schnitger, On the Power of Las Vegas for One-Way Communication Complexity, OBDD's and Finite Automata. *Information and Computation*, **169** (2001) 284-296.
12. J. Hromkovič, Relation Between Chomsky Hierarchy and Communication Complexity Hierarchy. *Acta Math. Univ. Com.*, Vol. **48-49** (1986), 311-317.
13. J. Hromkovič, G. Schnitger, Nondeterministic Computation with Limited Advise Bits. *Proc. 28th ACM STOC* (1996), 551-560.
14. J. Hromkovič, S. Seibert, J. Karhumäki, H. Klauck, G. Schnitger, Communication Complexity Method for Measuring Nondeterminism in Finite Automata. *Information and Computation*, **170** (2001), 1-16.
15. J. Hromkovič, Communication Complexity and Parallel Computing.Springer, 1997.
16. T. Jiang, E. McDowell, B. Ravikumar, The Structure and Complexity of Minimal NFA's over a Unary Alphabet. *International Journal of Foundation of Computer Science*, **2** (1991), 163-182.
17. T. Jiang, B. Ravikumar, Minimal NFA Problem is Hard. *SIAM Journal of Computing*, **22** (1993), 1117-1141.
18. G. Jirásková, Finite Automata and Communication Protocols. In: C. Martin-Vide, V. Mitrana (eds.) *Words, Sequences, Grammars, Languages: Where Biology, Computer Science, Linguistics and Mathematics Meet II* , to appear.
19. T. Kameda, P. Weiner, On the State Minimization of Nondeterministic Finite Automata. *IEEE Transaction on Computer* **C-19** (1970), 617-627.
20. E. Kushilevitz, N. Nisan, *Communication Complexity.* Cambridge University Press, 1997.
21. L. Lovász, Communication Complexity: A Survey. Techn. rep. CS-TR-204-89, Princeton University, 1989.
22. O. Matz, A. Potthoff, Computing Small Nondeterministic Finite Automata. In: U.H. Engberg, K.G. Larsen, A. Skou (eds.) *Proc. of the Workshop on Tools and Algorithms for Construction and Analysis of Systems*, BRICS Notes Series, (1995) 74-88.

23. N. Nisan, A. Wigderson, On Ranks vs. Communication Complexity. *Combinatorica* **15** (1995), 557-565.
24. C. Papadimitriou, M. Sipser, Communication Complexity. *Proc. 14th ACM STOC,* (1982), 196-200.
25. C. Pomerance, J.M. Robson, J. Shallit, Automaticity II: Descriptional Complexity in the Unary Case. *Theoretical Computer Science* **180** (1997) 181-201.
26. J. Shallit. Automaticity and Rationality. *Journal of Automata, Languages and Combinatorics* **5** (2000) 3, 255-268
27. J. Shallit, Y. Breitbart, Automaticity I: Properties of a Measure of Descriptional Complexity. *Journal of Computer and System Sciences* **53** (1996) 10-25.
28. C.D. Thompson, Area-time Complexity for VLSI. *Proc. 11th ACM STOC,* (1978), 81-88.
29. C.D. Thompson, A Complexity Theory for VLSI. Doctoral dissertation. CMU-CS-80-140, 131 p.,Computer Science Department, Carnegie-Mellon University, Pittsburgh, August 1980.
30. A. Yao, Some Complexity Questions Related to Distributed Computing. *Proc. 11th ACM STOC,* (1979), 209-213.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

95-11 *	M. Staudt / K. von Thadden: Subsumption Checking in Knowledge Bases
95-12 *	G.V. Zemanek / H.W. Nissen / H. Hubert / M. Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology
95-13 *	M. Staudt / M. Jarke: Incremental Maintenance of Externally Materialized Views
95-14 *	P. Peters / P. Szczurko / M. Jeusfeld: Business Process Oriented Information Management: Conceptual Models at Work
95-15 *	S. Rams / M. Jarke: Proceedings of the Fifth Annual Workshop on Information Technologies & Systems
95-16 *	W. Hans / St. Winkler / F. Sáenz: Distributed Execution in Functional Logic Programming
96-1 *	Jahresbericht 1995
96-2	M. Hanus / Chr. Prehofer: Higher-Order Narrowing with Definitional Trees
96-3 *	W. Scheufele / G. Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates
96-4	K. Pohl: PRO-ART: Enabling Requirements Pre-Traceability
96-5	K. Pohl: Requirements Engineering: An Overview
96-6 *	M. Jarke / W. Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools
96-7	O. Chitil: The ς-Semantics: A Comprehensive Semantics for Functional Programs
96-8 *	S. Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics
96-9	M. Hanus (Ed.): Proceedings of the Poster Session of ALP'96 — Fifth International Conference on Algebraic and Logic Programming
96-10	R. Conradi / B. Westfechtel: Version Models for Software Configuration Management
96-11 *	C. Weise / D. Lenzkes: A Fast Decision Algorithm for Timed Refinement
96-12 *	R. Dömges / K. Pohl / M. Jarke / B. Lohmann / W. Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models
96-13 *	K. Pohl / R. Klamma / K. Weidenhaupt / R. Dömges / P. Haumer / M. Jarke: A Framework for Process-Integrated Tools

96-14 * R. Gallersdörfer / K. Klabunde / A. Stolz / M. Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

96-15 * H. Schimpe / M. Staudt: VAREX: An Environment for Validating and Refining Rule Bases

96-16 * M. Jarke / M. Gebhardt, S. Jacobs, H. Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

96-17 M. Jeusfeld / T. X. Bui: Decision Support Components on the Internet

96-18 M. Jeusfeld / M. Papazoglou: Information Brokering: Design, Search and Transformation

96-19 * P. Peters / M. Jarke: Simulating the impact of information flows in networked organizations

96-20 M. Jarke / P. Peters / M. Jeusfeld: Model-driven planning and design of cooperative information systems

96-21 * G. de Michelis / E. Dubois / M. Jarke / F. Matthes / J. Mylopoulos / K. Pohl / J. Schmidt / C. Woo / E. Yu: Cooperative information systems: a manifesto

96-22 * S. Jacobs / M. Gebhardt, S. Kethers, W. Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

96-23 * M. Gebhardt / S. Jacobs: Conflict Management in Design

97-01 Jahresbericht 1996

97-02 J. Faassen: Using full parallel Boltzmann Machines for Optimization

97-03 A. Winter / A. Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

97-04 M. Mohnen / S. Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

97-05 * S. Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

97-06 M. Nicola / M. Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

97-07 P. Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

97-08 D. Blostein / A. Schürr: Computing with Graphs and Graph Rewriting

97-09 C.-A. Krapp / B. Westfechtel: Feedback Handling in Dynamic Task Nets

97-10 M. Nicola / M. Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

97-13 M. Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

97-14 R. Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

97-15 G. H. Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

98-01 * Jahresbericht 1997

98-02 S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes

98-03 S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

98-04 * O. Kubitz: Mobile Robots in Dynamic Environments

98-05 M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems

98-07 M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects

98-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen

98-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien

98-10 * M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases

98-11 * A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML

98-12 * W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web

98-13 K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation

99-01 * Jahresbericht 1998

99-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version

99-03 * R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager

99-04 M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing

99-07 Th. Wilke: CTL+ is exponentially more succinct than CTL

99-08 O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures

2000-01 * Jahresbericht 1999

2000-02 Jens Vöge / Marcin Jurdziński: A Discrete Strategy Improvement Algorithm for Solving Parity Games

2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach

2000-05 * Mareike Schoop: Cooperative Document Management

2000-06 * Mareike Schoop, Christoph Quix (Ed.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling

2000-07 * Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages

2000-08 Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations

2001-01 * Jahresbericht 2000

2001-02 Benedikt Bollig / Martin Leucker: Deciding LTL over Mazurkiewicz Traces

2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig / Martin Leucker / Michael Weber: Local Parallel Model Checking for the Alternation free μ-calculus

2001-05 Benedikt Bollig / Martin Leucker / Thomas Noll: Regular MSC languages

2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic

2001-07 Martin Grohe / Stefan Wöhrle: An Existential Locality Theorem

2001-08 Mareike Schoop / James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling

2001-09 Thomas Arts / Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs

2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures

2001-11 Klaus Indermark and Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung

2002-01 * Jahresbericht 2001

2002-02 Jürgen Giesl / Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

2002-03 Benedikt Bollig / Martin Leucker / Thomas Noll: Generalised Regular MSC Languages

2002-04 Jürgen Giesl / Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting

2002-05 Horst Lichter / Thomas von der Maßen / Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

* These reports are only available as a printed version.

Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.