International Journal of Foundations of Computer Science
Vol. 16, No. 5 (2005) 975–984

World Scientific
www.worldscientific.com

DESCRIPTIONAL COMPLEXITY OF NFA OF DIFFERENT AMBIGUITY

HING LEUNG*
Department of Computer Science
New Mexico State University
Las Cruces, NM 88003, U.S.A.
hleung@cs.nmsu.edu

Received 24 November 2004
Accepted 14 February 2005
Communicated by L. Ilie and D. Wotschke

In this paper, we study the tradeoffs in descriptional complexity of NFA (nondeterministic finite automata) of various amounts of ambiguity. We say that two classes of NFA are separated if one class can be exponentially more succinct in descriptional sizes than the other. New results are given for separating DFA (deterministic finite automata) from UFA (unambiguous finite automata), UFA from MDFA (DFA with multiple initial states) and UFA from FNA (finitely ambiguous NFA). We present a family of regular languages that we conjecture to be a good candidate for separating FNA from LNA (linearly ambiguous NFA).

Keywords: Finite automata; nondeterminism; ambiguity; descriptional complexity.

1991 Mathematics Subject Classification: 68Q19, 68Q45

1. Introduction

The tradeoffs in descriptional complexity of finite automata have been widely studied since the 1970's ([11], [13]). In this paper, we study the tradeoffs in descriptional complexity of nondeterministic finite automata (NFA) of various amounts of ambiguity. A recent survey on the topic can be found at [2].

In this paper, we allow nondeterministic finite automata (NFA) to have multiple initial states. Given an NFA M, we define the ambiguity of a string w to be the number of different accepting paths for w in M. An NFA is said to be k-ambiguous if every string in the language is accepted with at most k different accepting computations. An unambiguous NFA (UFA) is a 1-ambiguous NFA. An NFA is said to be finitely ambiguous (FNA) if the NFA is k-ambiguous for some positive integer k.

*The research was done while the author was visiting University of Frankfurt, Germany, and was supported by the Alexander von Humboldt foundation and NSF MII grant EIA-0220590.

There is a special class of FNA called deterministic finite automata with multiple initial states (MDFA) [7] which is an NFA with deterministic transition logic. An MDFA is k-ambiguous where k is the number of starting states.

An NFA is polynomially ambiguous (PNA) if there exists a polynomial p such that every string x in the language is accepted with at most $p(|x|)$ accepting computations. When p is a linear function, a PNA is also called a linearly ambiguous NFA (LNA). Given an NFA of k states, any input string of length n can have at most k^n different accepting computations. Thus, it follows that every NFA is exponentially ambiguous (ENA).

Let C_1 and C_2 be two classes of NFA of different ambiguity. We say that C_1 can be polynomially converted to C_2 (written $C_1 \leq_P C_2$) if there exists a polynomial p such that for any finite automaton in C_1 with n states, we can find an equivalent finite automaton in C_2 with at most $p(n)$ states. C_1 is said to be polynomially related to C_2 (written $C_1 =_P C_2$) if $C_1 \leq_P C_2$ and $C_2 \leq_P C_1$. C_1 is said to be separated from C_2 if $C_1 \neq_P C_2$. We write $C_1 <_P C_2$ if $C_1 \leq_P C_2$ and $C_1 \neq_P C_2$.

It is immediate that DFA $\leq_P$ UFA, DFA $\leq_P$ MDFA, UFA $\leq_P$ FNA, MDFA $\leq_P$ FNA, FNA $\leq_P$ PNA, and PNA $\leq_P$ ENA.

The following results are known[a]: DFA $<_P$ UFA ([13], [14], [12], [8]), DFA $<_P$ MDFA ([1], [15], [7], [6]), UFA $<_P$ FNA ([13], [12]) and PNA $<_P$ ENA ([9], [5]).

In this paper, we present interesting new families of regular languages for separating DFA from UFA, UFA from MDFA, and UFA from FNA.

It is an open problem whether FNA $<_P$ PNA. The problem is very difficult. Previous attempts for separating FNA from PNA have failed [5]. More exactly, two families of regular languages given in [12] and [4] that are conjectured to separate FNA from PNA are found to be easy by M. Jurdzinski [5] for FNA.

We present a family of regular languages that we conjecture to be a good candidate for separating FNA from LNA.

2. Tradeoffs between DFA and UFA

Schmidt [13] gave a family of n-state UFA such that the smallest equivalent DFA require $2^{c\sqrt{n}}$ states.

Consider the regular language $(0+1)^*0(0+1)^{n-2}$. It is well known ([11], [14]) since the 1970's that there is an NFA of n states for the language, whereas the smallest equivalent DFA has 2^{n-1} states.

Leiss [8] showed that there is a family of n-state UFA such that the equivalent DFA require exactly 2^n states. Note that exactly half of the states are designated as starting states.

We have discovered a family[b] of UFA $\mathcal{U}_n$ that demonstrates the largest tradeoffs between UFA and DFA. See Figure 1 for the pictures of the UFA $\mathcal{U}_9$. Unlike the

[a]Indeed, there are more known results than what we have listed here. See [2].

[b]Another family of UFA U_n that demonstrates the largest tradeoffs between UFA and DFA is given in [10], where U_n has $3n$ states and the smallest equivalent DFA has 2^{3n} states.

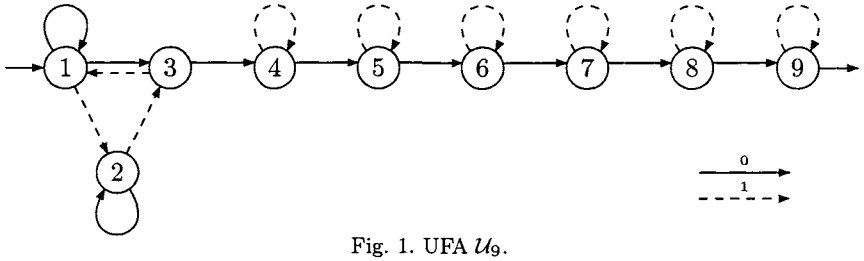

Fig. 1. UFA $\mathcal{U}_9$.

construction by Leiss, $\mathcal{U}_n$ has one start state (State 1) and one final state (State n). Note that $\mathcal{U}_n$ has n states and is defined for $n \geq 3$. As the reversal of $\mathcal{U}_n$ is a DFA, we see that $\mathcal{U}_n$ is indeed a UFA.

Theorem 1. *The smallest equivalent DFA for $\mathcal{U}_n$ has 2^n states, where $n \geq 3$.*

Proof. Since the reversal of $\mathcal{U}_n$ is a DFA, all subsets of states are distinguishable[c]. The reasons are as follows: Let the final state of $\mathcal{U}_n$ be q_f. Consider two different subsets of states Q_1 and Q_2. Without loss of generality, suppose $q \in Q_1$ and $q \notin Q_2$. Since the reversal of $\mathcal{U}_n$ is a DFA, there exists a string w that takes the reversal of $\mathcal{U}_n$ from its starting state q_f to q in a deterministic path. Thus, in $\mathcal{U}_n$, q_f is reached from Q_1 on processing w, whereas q_f is not reached from Q_2 on processing w. That is, Q_1 and Q_2 are distinguished.

It remains to show that all subsets of states are reachable from the starting state. With the string 110^{n-2}, we reach the empty subset of states from the starting state.

It is easy to reach non-empty subsets of states 1, 2 and 3. We reach the subsets $\{1\}, \{2\}, \{3\}, \{1,3\}, \{1,2\}, \{2,3\}, \{1,2,3\}$ by processing strings $\epsilon, 1, 11, 0, 01, 011, 010$ respectively from the starting state.

Induction hypothesis: Let $k \geq 3$. For all $Q \subseteq \{1, 2, \ldots, k\}$, Q is reachable.

Induction step: Consider Q such that the largest numbered state in Q is $k+1$.

Case 1: $Q \cap \{1,3\} = \emptyset$ **or** $\{2\}$ **or** $\{1,3\}$ **or** $\{1,2,3\}$**.** The subset Q can be reached by processing a symbol 0 from Q' that is guaranteed to be reachable by the induction hypothesis. Formally, Q' is defined as $\{1 \mid 1 \in Q\} \cup \{2 \mid 2 \in Q\} \cup \{i \mid 4 \leq i+1 \in Q\}$.

Case 2: $Q \cap \{1,2,3\} = \{3\}$**.** The subset Q can be reached by processing a 1 from $Q' = Q \cup \{2\} \setminus \{3\}$ which is reachable since it satisfies the condition for Case 1.

Case 3: $Q \cap \{1,2,3\} = \{1\}$**.** The subset Q can be reached by processing a 1 from $Q' = Q \cup \{3\} \setminus \{1\}$ which is reachable since it satisfies the condition for Case 2.

[c]To reason that a DFA is a minimum in size, Myhill-Nerode theorem states that any two states q and q' must be distinguishable in that there exists a string w such that $\delta(q, w) \in F$ iff $\delta(q', w) \notin F$ where δ is the transition function and F is the set of final states. In our case, the DFA is obtained by subset construction. Thus, we need to show that all subsets of states are distinguishable.

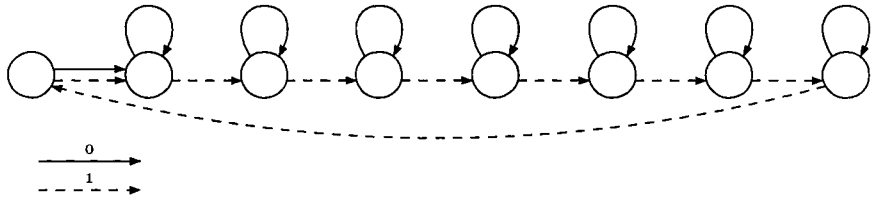

Fig. 2. Veloso and Gill's MDFA (starting and final states not shown).

Case 4: $Q \cap \{1,2,3\} = \{1,2\}$. The subset Q can be reached by processing a 1 from $Q' = Q \cup \{3\} \backslash \{2\}$ which is reachable since it satisfies the condition for Case 1.

Case 5: $Q \cap \{1,2,3\} = \{2,3\}$. The subset Q can be reached by processing a 1 from $Q' = Q \cup \{1\} \backslash \{3\}$ which is reachable since it satisfies the condition for Case 4. □

3. Tradeoffs between UFA and MDFA/FNA

In [13], it is proved that there exists a family of n-state FNA whose smallest equivalent UFA has $2^{\Omega(\sqrt{n})}$ states. To prove a lower bound for the size of a UFA, Schmidt [13] introduced the following result, which can be proved using communication complexity techniques [5].

Theorem 2 (Schmidt) *Given a regular language L and strings x_i, y_i for $i = 1, \ldots, n$, let M be a matrix such that $M[x_i, y_j] = 1$ if $x_i y_j \in L$, and 0 otherwise. Then any UFA for L has at least rank(M) states.*

Proof. (Sketch) Let Q be the set of states of a UFA A for L. We define a matrix M' with rows indexed by $q \in Q$ and columns indexed by y_i for $i = 1, \ldots, n$ such that $M'[q, y_i] = 1$ if from q we can process y_i and arrive at a final state of A, and 0 otherwise. Consider x_i and the i-th row of M. Let Q' be the set of states reached in A by processing x_i from the starting states. One can show that the i-th row of M is the sum of rows of M' indexed by states in Q'. Note that if $M'[q, y_j] = 1$ for some $q \in Q'$, it follows that $M'[q', y_j] = 0$ for all $q \neq q' \in Q'$; otherwise, A is not a UFA. That is, each row in M is a linear combination of the rows in M'. Thus, $\text{rank}(M) \leq \text{rank}(M') \leq |Q|$, where $|Q|$ is the number of rows in M'. □

In this section, we are going to prove the largest possible tradeoff between UFA and FNA. That is, for some family of n-state FNA, we will show that the smallest equivalent UFA has $2^n - 1$ states.

Consider the finite automaton given in Figure 2. By picking all states as starting states and an arbitrary state as final state, we obtain Veloso and Gill's MDFA [15]. It was proved that the smallest equivalent DFA has $2^n - 1$ states where n is the number of states of the corresponding MDFA. Specifically, all $2^n - 1$ nonempty

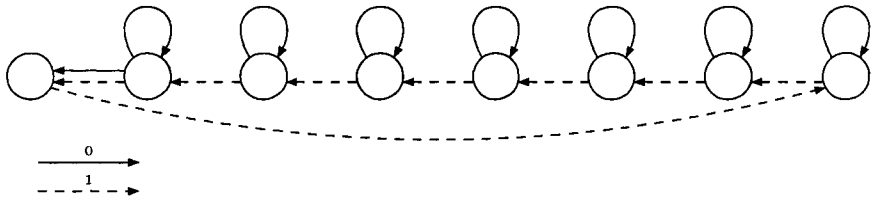

Fig. 3. Leiss's NFA (Starting and final states not shown).

subsets of states are reachable and distinguishable. To reach a nonempty subset of states, we need to process a string x from the set of starting states. Each nonempty subset of states corresponds to a different string x. Let x_i, for $i = 1, 2, \ldots, 2^n - 1$, be the strings that correspond to the $2^n - 1$ different nonempty subsets of states.

Interestingly, the reversal of Veloso and Gill's design was also considered by Leiss [8]. See Figure 3 for Leiss's NFA. An arbitrary state is picked as the final state. Unlike Veloso and Gill's MDFA, not all states are picked as starting states. Alternate (non-neighboring) states are selected as starting states. Thus, half of the states are starting states. Observe that the NFA is indeed a UFA as the reversal is a DFA. It can be shown [8] that the smallest equivalent DFA has 2^n states where n is the number of states of Leiss's NFA. That is, all 2^n subsets of states are reachable and distinguishable. As in the case for Veloso and Gill's MDFA, we let z_i, for $i = 1, 2, \ldots, 2^n - 1$, be the strings that correspond to the $2^n - 1$ different nonempty subsets of states.

We combine the two designs by Veloso and Gill, and by Leiss. Consider Veloso and Gill's MDFA. All states are starting states. We define the final states to consist of alternate non-neighboring states. We call the resulting MDFA M_n.

Theorem 3. *The smallest UFA equivalent to the MDFA M_n has $2^n - 1$ states.*

Proof. Define y_i to be the reversal of the string z_i, for $i = 1, 2, \ldots, 2^n - 1$. As in Theorem 2, we define M with respect to the language of M_n and based on the strings x_i's and y_i's. The matrix M can be re-stated as a matrix indexed by nonempty subsets of states such that entry $[Q_1, Q_2]$ has the value 1 if $Q_1 \cap Q_2 \neq \emptyset$, otherwise the entry has the value 0. It has been shown in [9] that M has rank $2^n - 1$. Thus, by Theorem 2, the smallest UFA equivalent to M_n has $2^n - 1$ states. □

By reversing the design of M_n (reverse all transitions, all states are final, alternate states are starting), we obtain an FNA M'_n. As a corollary of Theorem 3, we have

Corollary 4. *The smallest UFA equivalent to the FNA M'_n has $2^n - 1$ states.*

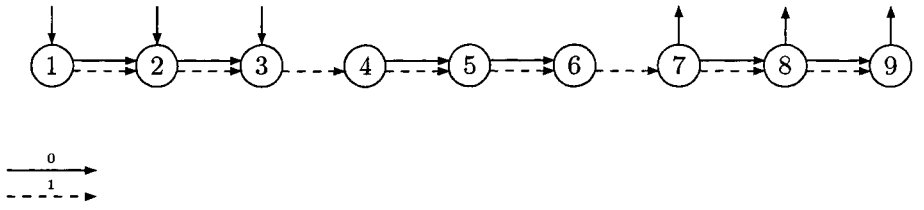

Fig. 4. MDFA A_3.

Note that the smallest equivalent DFA for M_n has $2^n - 1$ states, whereas the smallest equivalent DFA for M'_n has 2^n states.

Next, we consider the tradeoffs in descriptional complexity between UFA and MDFA for finite languages.

Consider the MDFA A_3 given in Figure 4. In general, A_n has $3n$ states. The first n states are starting states, whereas the last n states are final states. Note that A_n is reverse-symmetric, and recognizes a finite language.

To prove a lower bound on the size of an equivalent UFA for A_n using the technique given in Theorem 2, we define strings x_i's and y_i's to be the set of 2^n strings over $\{0,1\}$ of length n.

Consider a string $x \in \{0,1\}^n$. Let $x = a_1 a_2 \ldots a_n$ where $a_i \in \{0,1\}$. Suppose a_j is 1. Initially, state $n - j + 1$ is on. After processing the first $j - 1$ symbols of x, state n is on. Next, we process the j-th symbol a_j which is 1. Thus, state $n+1$ is on after processing a_j. There are $n - j$ symbols $a_{j+1} \ldots a_n$ to be processed. Therefore, the state $n + (n - j) + 1$ is on after processing x. That is, $a_1 = 1$ implies state $2n$ is on after processing x, $a_2 = 1$ implies state $2n - 1$ is on after processing x,...etc. Finally, $a_n = 1$ implies state $n + 1$ is on after processing x. As the MDFA A_n is reverse-symmetric, similar behaviors are observed when we process a string $y \in \{0,1\}^n$ backwards starting from the final states. We define M based on x_i's and y_i's. The matrix M can be re-stated as a matrix indexed by subsets of the states set $\{n+1, n+2, \ldots, 2n\}$ such that entry $[Q_1, Q_2]$ has the value 1 if $Q_1 \cap Q_2 \neq \emptyset$, otherwise the entry has the value 0. Again, the rank of M is $2^n - 1$. Therefore, we obtain the next theorem.

Theorem 5. *The smallest UFA equivalent to the MDFA A_n has at least $2^n - 1$ states. Note that A_n recognizes a finite language, and has $3n$ states.*

As the strings x_i's and y_i's are all of length n, we deduce that it also requires at least $2^n - 1$ states for a UFA to recognize the language $L(A_n) \cap \{0,1\}^{2n}$, which we denote by L_n. Figure 5 shows an MDFA for L_3. The design can be further reduced slightly in size by combining the starting states and the final states into one starting state and one final state respectively. In general, there is an MDFA for L_n with $O(n^2)$ states.

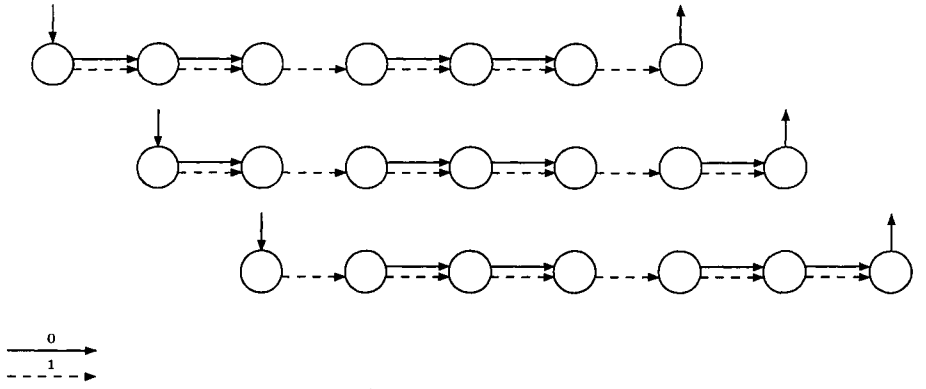

Fig. 5. MDFA for L_3.

4. Tradeoffs between FNA and LNA

It is an open problem whether there can be exponential tradeoffs in descriptional complexity between FNA and PNA. In this section, we define a language family and conjecture that it is good candidate for separating FNA from LNA.

Suppose there are n registers. Each register holds a value of either 0 or 1 ('off' or 'on'). Initially, register 1 is on. All other registers are off. Consider an instruction

`Copy` i `to` j

Executing the instruction will copy the current value of register i to register j. In short, the instruction is given as $C_{i,j}$.

We define an input string to consist of a sequence of copy instructions. As copying register i to itself is a dummy instruction, we assume that $C_{i,i}$ is not allowed. As there are $n(n-1)$ possible copy instructions, the input alphabet has $O(n^2)$ letters.

An example input is $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}$. We say that an input is in the language of some-register-on if some register is on after the sequence of copy instructions have been performed.

Consider the example input given before. Initially, register 1 is on. The first copy instruction $C_{1,4}$ will turn register 4 to on. The next instruction $C_{4,2}$ will turn on register 2. The next instruction $C_{3,1}$ sets register 1 to off as register 3 is off. Next, instruction $C_{4,3}$ turns on register 3. The last instruction $C_{1,2}$ turns register 2 to off. Thus, after all copy instructions are performed, registers 3 and 4 are on whereas registers 1 and 2 are off. Since not all registers are off, we conclude that the input belongs to the language of some-register-on.

We define an n-state FNA for the language of some-register-on. State 1 is an initial state. The design intuition is that state i is loaded when register i is on. With

respect to the input symbol $C_{i,j}$, there are transitions going from state i to state j, and transitions going from state k to state k where $k \neq j$. All states are final states. To see that the NFA defined is an FNA, we reverse the NFA construction. The logic after reversing all transitions is deterministic except that we have n starting states. Thus, we have an MDFA when the NFA is reversed. Hence, the given NFA is an FNA.

We modify the language of some-register-on. Instead of accepting a string when some register is on, we accept a string only if the register that we query at the end of the input is on. That is, the end of an input is augmented by a query instruction.

`Assert: Register` i `is on`

An input is accepted if the register i queried is on. In short, the query instruction is denoted as Q_i.

We extend the previous example input by a query Q_3. The example input becomes $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_3$. Since register 3 is on after the copy instructions are performed, the input is accepted as the query is about register 3. If we query about register 1 at the end of the input as in $C_{1,4}C_{4,2}C_{3,1}C_{4,3}C_{1,2}Q_1$, then the input is not accepted since register 1 is off.

To handle the newly added query feature, we modify the FNA for the language of some-register-on. We introduce a new state called f, which is the only final state. New transitions are added: from each state i, on processing Q_i, it will go to state f. The resulting $(n+1)$-state NFA is a UFA. We can see this as the reversal of the NFA is a DFA.

We can prove that the $(n+1)$-state NFA is indeed a minimal NFA for the language with a query at the input end. A sketch of the proof is as follows: Pick $x_1 = \epsilon, x_2 = C_{1,2}C_{3,1}, x_i = C_{1,i}C_{2,1}$ for $i = 3, 4, .., n$, $y_1 = Q_1$, $y_2 = Q_2$, ..., $y_n = Q_n$. Using the fooling set method [3], we need at least n states. But these n states cannot be final states. We still need one more final state. Thus, we need at least $n+1$ states for an NFA. As the smallest NFA requires $n+1$ states, we conclude that the smallest UFA also requires $n+1$ states. Therefore, our construction gives a smallest UFA for the language.

Hromkovič et al. [5] has made a conjecture about minimal UFA computations. Specifically, it is conjectured that if a computation branches into two computations, then at least one computation is completely deterministic. However, a close look at the computations of our minimal UFA shows that the conjecture is not true.

Next, we further extend the language by allowing multiple queries. That is, queries can be made at any point of the input string. The input is accepted if at least one query is answered positively.

An example input is $C_{1,4}C_{4,2}Q_3C_{3,1}C_{4,3}Q_1C_{1,2}$. When the first query Q_3 is made, registers 1, 2 and 4 are on but register 3 is off. When the second query Q_1 is made, registers 2, 3 and 4 are on but register 1 is off. The input is not accepted as both queries are answered negatively. On the other hand, the example input $C_{1,4}C_{4,2}Q_2C_{3,1}C_{4,3}Q_1C_{1,2}$ is accepted.

We add new transitions to the previous UFA to handle multiple queries. From state f, on processing any input symbol, there is a transition going back to f. From any state i ($i = 1, 2, \ldots, n$), on query Q_j where $j \neq i$, we introduce transitions that go back to i.

At any moment during the processing of an input, at most one computation arrives at each of the state i for $i = 1, 2, \ldots, n$. A query may cause a path to move into the final state f. Thus, the number of accepting paths is bounded by the number of queries. On the other hand, the number of accepting computations could be the same as the number of queries which can illustrated by the processing of the following example input $C_{1,2}Q_2C_{1,2}Q_2C_{1,2}Q_2C_{1,2}Q_2$. Therefore, with the new transitions, we have a $(n+1)$-state LNA. We conjecture that any FNA for the language with multiple queries has at least 2^n states, which is the size of the smallest DFA.

Interestingly, the language with multiple queries is easy for two-way DFA. That is, there is a small two-way DFA for the language that employs depth first search processing.

Acknowledgements

The author would like to thank Lane Hemaspaandra for valuable discussions.

References

[1] A. Gill and L. T. Kou, Multiple-entry finite automata, *J. Comput. System Sci.* **9**(1974) 1–19.

[2] J. Goldstine, M. Kappes, C. M. R. Kintala, H. Leung, A. Malcher and D. Wotschke, Descriptional complexity of machines with limited resources, *J. Univ. Comput. Sci.* **8**(2002) 193–234.

[3] I. Glaister and J. Shallit, A lower bound technique for the size of nondeterministic finite automata, *Inform. Process. Lett.* **59**(1996) 75–77.

[4] J. Hromkovič, J. Karhumäki, H. Klauck, S. Seibert and G. Schnitger, Measures of nondeterminism in finite automata, in *Proc. ICALP'00*, Lecture Notes in Computer Science, vol. 1853, (Springer-Verlag, Berlin, 2000), pp. 199–210.

[5] J. Hromkovič, J. Karhumäki, H. Klauck, G. Schnitger and S. Seibert, Communication complexity method for measuring nondeterminism in finite automata, *Inform. and Comput.* **172**(2002) 202–217.

[6] M. Holzer, K. Salomaa and S. Yu, On the state complexity of k-entry deterministic finite automata, *J. Autom. Lang. Comb.* **6**(2001) 453–466.

[7] M. Kappes, Descriptional complexity of deterministic finite automata with multiple initial states, *J. Autom. Lang. Comb.* **5**(2000) 269–278.

[8] E. Leiss, Succinct representation of regular languages by Boolean automata, *Theoret. Comput. Sci.* **13**(1981) 323–330.

[9] H. Leung, Separating exponentially ambiguous finite automata from polynomially ambiguous finite automata, *SIAM J. Comput.* **27**(1988) 1073–1082.

[10] H. Leung, Descriptional complexity of NFAs of different ambiguity, in *Proc. DCFS 2004*. (University of Western Ontario Computer Science Technical Report 619, 2004), pp. 98–107.

[11] A. R. Meyer and M. Fischer, Economy of description by automata, grammars, and formal systems, in *IEEE Twelfth Annual Symposium on Switching and Automata Theory*, (IEEE, 1971), pp. 188–191.

[12] B. Ravikumar and O. H. Ibarra, Relating the type of ambiguity of finite automata to the succinctness of their representation, *SIAM J. Comput* **18**(1989) 1263–1282.

[13] E. M. Schmidt, *Succinctness of descriptions of context-free, regular and finite languages*, PhD Thesis, Cornell University, Ithaca, NY (1978).

[14] R. E. Stearns and H. B. Hunt, III, On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata, *SIAM J. Comput.* **14**(1985) 598–611.

[15] P. A. S. Veloso and A. Gill, Some remarks on multiple-entry finite automata, *J. Comput. System. Sci.* **18**(1979) 304–306.