RAIRO-Inf. Theor. Appl. 40 (2006) 501-510

DOI: 10.1051/ita:2006033

# NOTE ON THE COMPLEXITY OF LAS VEGAS AUTOMATA PROBLEMS\*

## Galina Jirásková<sup>1</sup>

**Abstract.** We investigate the complexity of several problems concerning Las Vegas finite automata. Our results are as follows. (1) The membership problem for Las Vegas finite automata is in NL. (2) The nonemptiness and inequivalence problems for Las Vegas finite automata are NL-complete. (3) Constructing for a given Las Vegas finite automaton a minimum state deterministic finite automaton is in NP. These results provide partial answers to some open problems posed by Hromkovič and Schnitger [Theoret. Comput. Sci. **262** (2001) 1–24)].

Mathematics Subject Classification. 68Q19, 68Q17.

#### 1. Introduction

A Las Vegas finite automaton is a probabilistic finite automaton (with a single initial state) whose states are divided into three disjoint groups: accepting states, rejecting states, and neutral ("I don't know") states. For any input string, the automaton never errs and the probability of giving the answer "I don't know" is at most 1/2.

Las Vegas finite automata (LVFAs) were introduced by Hromkovič and Schnitger [4], where the authors showed that there is at most a quadratic difference between deterministic finite automata and Las Vegas finite automata. They proved the result by using a reduction to one-way Las Vegas communication protocols. A direct proof of the above result, based on information theory, was given by Hirvensalo and Seibert in [2]. The relation between the power of determinism, Las Vegas, and nondeterminism for two-way finite automata was studied by

© EDP Sciences 2006

Keywords and phrases. Las Vegas finite automata, deterministic and nondeterministic finite automata, computational complexity.

<sup>\*</sup> Research supported by the VEGA grant No. 2/6089/26.

 $<sup>^1</sup>$  Mathematical Institute, Slovak Academy of Sciences, Grešákova 6, 040 01 Košice, Slovakia; jiraskov ${\tt 0saske.sk}$ 

Hromkovič and Schnitger in [5]. In the paper, the authors stated several problems concerning Las Vegas finite automata:

"... One-way Las Vegas finite automata work in linear time and so they are a reasonable data structure for regular languages if one wants to decide the membership problem. But the efficiency of the use of LVFAs for regular language representation depends on the kind of manipulations one would like to realize with the representation. How hard is to find a minimal LVFA? How hard is to construct the minimal DFA for a given LVFA? The straightforward approach uses the construction of an equivalent DFA for a given NFA, and so the space complexity of this algorithm is exponential. Does there exist a more efficient way? How hard is it to decide the equivalence problem for LVFAs?"

In the present paper, we give partial answers to some of the above questions. Our results are as follows.

- (1) The membership problem for Las Vegas finite automata is in NL.
- (2) The nonemptiness and inequivalence problems for Las Vegas finite automata are NL-complete.
- (3) Constructing for a given Las Vegas finite automaton a minimum state deterministic finite automaton is in NP.

We recall that the nonemptiness problem for DFAs or NFAs, and the inequivalence problem for DFAs are known to be NL-complete [10,11], the equivalence and minimization for DFAs are NL-complete as well [1], while the inequivalence problem for NFAs is PSPACE-complete [16]. Further results on the complexity of decision problems for finite-state automata can be found in [6,8,9,18,19].

The paper consists of five sections, including this introduction. The next section contains basic definitions and notations used throughout the paper. In Section 3, we investigate the complexity of the membership, nonemptiness, and inequivalence problems for Las Vegas Finite automata. Section 4 studies the question of how hard is it to construct the minimal deterministic finite automaton equivalent to a given Las Vegas finite automaton. The last section contains concluding remarks and open problems.

# 2. Preliminaries

In this section, we recall some basic definitions and notations. For further details, we refer to [2, 4, 15].

Let  $\Sigma$  be an alphabet and  $\Sigma^*$  the set of all strings over  $\Sigma$  including the empty string  $\varepsilon$ . The length of a string w is denoted by |w| and the ith symbol of a string w is denoted by w[i]. A language over an alphabet  $\Sigma$  is a subset of  $\Sigma^*$ . Given a language  $L \subseteq \Sigma^*$ ,  $L^c$  denotes its complement, i.e., the set  $\{w \in \Sigma^* \mid w \notin L\}$ . The cardinality of a set A is denoted by |A| and its power-set by  $2^A$ .

A (row) vector is *stochastic* if all its entries are greater than or equal to zero and sum to 1. A matrix is stochastic if all its row vectors are stochastic.

A deterministic finite automaton (DFA) is a 5-tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ , where Q is a finite nonempty set of states,  $\Sigma$  is a finite alphabet,  $\delta: Q \times \Sigma \to Q$  is the transition function,  $q_0 \in Q$  is the initial state, and  $F \subseteq Q$  is the set of accepting states. The transition function  $\delta$  can be extended to the domain  $Q \times \Sigma^*$  in the natural way. A string w in  $\Sigma^*$  is accepted by the DFA  $\mathcal{A}$  if the state  $\delta(q_0, w)$  is an accepting state of the DFA  $\mathcal{A}$ .

A nondeterministic finite automaton (NFA) is a 5-tuple  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ , where  $Q, \Sigma, q_0$ , and F are defined as for a DFA, and  $\delta : Q \times \Sigma \to 2^Q$  is the transition function which can be naturally extended to the domain  $Q \times \Sigma^*$ . A string w in  $\Sigma^*$  is accepted by the NFA  $\mathcal{A}$  if the set  $\delta(q_0, w)$  contains an accepting state of the NFA  $\mathcal{A}$ . Clearly, the NFA  $\mathcal{A}$  accepts a string w of length m if a sequence of states  $s_0, s_1, \ldots, s_m$  exists in Q such that  $s_0 = q_0, s_i \in \delta(s_{i-1}, w[i])$  for  $i = 1, 2, \ldots, m$ , and  $s_m \in F$ .

A Las Vegas finite automaton (LVFA) is a 6-tuple  $\mathcal{A} = (Q, \Sigma, M, q_0, A, R)$ , where  $Q, \Sigma$ , and  $q_0$  are as above, M is a function from  $\Sigma$  into the set of all  $(|Q| \times |Q|)$ -dimensional stochastic matrices, A and R are disjoint subsets of Q, A is the set of accepting states, R is the set of rejecting states. The states that are neither accepting nor rejecting are called neutral. For any symbol a in  $\Sigma$ , the matrix M(a) is a stochastic matrix whose rows and columns are indexed by the states of Q (in some fixed order) and the value M(a)[p,q] is the probability that the LVFA  $\mathcal A$  moves from state p to state q after reading symbol a. A computation of the LVFA  $\mathcal A$  on a string w of length m is a sequence of states  $s_0, s_1, \ldots, s_m$  such that  $s_0 = q_0$ , and  $M(w[i])[s_{i-1}, s_i] > 0$  for  $i = 1, 2, \ldots, m$ . The probability of a computation of the LVFA  $\mathcal A$  on a string w is the product of the transition probabilities along the path of the computation. Moreover, the following two conditions must hold:

- (i) for no input string there exist two computations of the LVFA  $\mathcal{A}$  ending up in an accepting and rejecting state, respectively, and
- (ii) for any input string the LVFA  $\mathcal{A}$  reaches a neutral state with the probability at most 1/2.

A string w in  $\Sigma^*$  is accepted by the LVFA  $\mathcal{A}$  if there is a computation of  $\mathcal{A}$  on w ending up in an accepting state. Note that if a string is accepted by an LVFA, then the probability of reaching an accepting state is at least 1/2 and the probability of reaching a rejecting state is zero, and if a string is not accepted by an LVFA, then the probability of reaching a rejecting state is at least 1/2 and the probability of reaching an accepting state is zero.

The language accepted by a finite automaton  $\mathcal{A}$  (deterministic, nondeterministic, or Las Vegas), denoted  $L(\mathcal{A})$ , is the set of strings accepted by the automaton  $\mathcal{A}$ . Two automata are said to be *equivalent* if they accept the same language.

Any DFA can be easily converted to an equivalent LVFA with the same state set, the same accepting and rejecting states (and no neutral states), and transition probabilities 1 for all transitions of the DFA and zero otherwise. On the other hand, any LVFA can be converted to an equivalent NFA by ignoring transition

probabilities and by adding the neutral states to the set of rejecting states. Thus, the LVFAs accept the family of regular languages.

A DFA  $\mathcal{A}$  is called *minimal* if all DFAs that are equivalent to  $\mathcal{A}$  have at least as many states as the DFA  $\mathcal{A}$ . By a well-known result, each regular language has a unique minimal DFA, up to isomorphism.

In the rest of the paper, NL denotes the class of problems that can be solved by a nondeterministic Turing machine operating in  $O(\log n)$  space, NL-completeness is with respect to log-space reductions, and NP denotes the class of problems that can be solved by a nondeterministic Turing machine operating in polynomial time. Finally, we assume that all stochastic matrices (given at the input) consist of rational numbers.

### 3. The complexity of decision problems for LVFAs

In this section, we investigate the complexity of the following problems for Las Vegas finite automata: (i) membership: Given an LVFA  $\mathcal{A}$  with the input alphabet  $\Sigma$  and a string w in  $\Sigma^*$ , is  $w \in L(\mathcal{A})$ ? (ii) nonemptiness: Given an LVFA  $\mathcal{A}$ , is  $L(\mathcal{A}) \neq \emptyset$ ? (iii) inequivalence: Given LVFAs  $\mathcal{A}_1$  and  $\mathcal{A}_2$ , is  $L(\mathcal{A}_1) \neq L(\mathcal{A}_2)$ ?

The first theorem shows that the membership problem for Las Vegas finite automata is in NL. The next two theorems show that the nonemptiness and inequivalence problems for Las Vegas finite automata are NL-complete, so are of the same complexity as the corresponding problems for deterministic finite automata (cf. [10,11]).

**Theorem 1.** The membership problem for Las Vegas finite automata is decidable in nondeterministic logarithmic space.

*Proof.* The nondeterministic algorithm for the membership problem starts at the initial state of the given LVFA. Then it nondeterministically guesses the steps of an accepting computation on the given input string. If its guess is successfull, then it accepts, otherwise it rejects:

```
Algorithm for LVFA membership problem

Input: An LVFA \mathcal{A} = (Q, \Sigma, M, q_0, A, R) and a string w in \Sigma^*

Task: Find whether w \in L(\mathcal{A})

s := q_0; \ m := |w|;

for i := 1 to m do

guess a state q \in Q such that M(w[i])[s, q] > 0;
s := q;
od;

if s \in A then accept else reject;
```

Clearly, the algorithm runs in log space.

**Theorem 2.** The nonemptiness problem for Las Vegas finite automata is NL-complete.

*Proof.* The language accepted by an LVFA of n states is nonempty if and only if the LVFA accepts a string of length less than n. The following nondeterministic algorithm starts with the initial state (given at the input) and a counter which is initialized to the number of states of the given LVFA. At each step it decrements the counter, guesses a symbol from the input alphabet and a state which can be reached from the current state after reading the guessed symbol with nonzero probability. The algorithm concludes when either the counter reaches zero or the last state it guessed is accepting:

```
Input: An LVFA \mathcal{A} = (Q, \Sigma, M, q_0, A, R)

Task: Find whether L(\mathcal{A}) \neq \emptyset

s := q_0; n := |Q|;

while n > 0 do

if s \in A then accept and halt;

n := n - 1;

guess a symbol a \in \Sigma;

guess a state q \in Q such that M(a)[s, q] > 0;
```

Algorithm for LVFA nonemptiness problem

The algorithm only needs to store s, n, a, q, and a pointer to  $\mathcal{A}$ , so it uses log space.

To prove NL-hardness of the problem, we give a log space reduction from NL-complete problem PATH [15]: Given an oriented graph G and its two vertices s and t, has the graph G a directed path from s to t?

Let G be an oriented graph with the set of vertices V and let s and t be two vertices in V. We construct an LVFA A in log space, such that the language L(A) is nonempty if and only if the graph G has an oriented path from s to t.

Define an LVFA  $\mathcal{A} = (Q, \Sigma, M, q_0, A, R)$ , where  $Q = \Sigma = V$ ,  $q_0 = s, A = \{t\}$ ,  $R = V \setminus \{t\}$ , and for any p and q in Q and any a in  $\Sigma$ ,

if (p, a) is an edge of the graph G, then

$$M(a)[p,q] = \left\{ \begin{array}{ll} 1, & \text{if } q = a, \\ 0, & \text{if } q \neq a, \end{array} \right.$$

otherwise

s := q;

od
reject;

$$M(a)[p,q] = \left\{ \begin{array}{ll} 1, & \text{if } q = p, \\ 0, & \text{if } q \neq p, \end{array} \right.$$

i.e., the LVFA  $\mathcal{A}$  moves from state p to state a on reading symbol a with probability 1 if (p, a) is an edge of the graph G, otherwise it moves from state p to state p on reading symbol a with probability 1 (in fact,  $\mathcal{A}$  is a deterministic automaton).

The reduction operates in log space and, clearly, the language L(A) is nonempty if and only if the graph G has an oriented path from s to t.

Thus our proof is complete.

We now turn our attention to the inequivalence problem for Las Vegas finite automata. The next lemma shows that given two LVFAs, we can easily construct a nondeterministic finite automaton for the symmetric difference of the languages accepted by the given LVFAs.

**Lemma 1.** Let  $L_1$  and  $L_2$  be regular languages accepted by Las Vegas finite automata of m and n states, respectively. Then there is an mn-state nondeterministic finite automaton accepting the language  $(L_1 \cap L_2^c) \cup (L_1^c \cap L_2)$ .

Proof. Let  $A_1 = (Q_1, \Sigma, M_1, q_{01}, A_1, R_1)$  and  $A_2 = (Q_2, \Sigma, M_2, q_{02}, A_2, R_2)$  be Las Vegas finite automata accepting languages  $L_1$  and  $L_2$ , respectively. Construct the nondeterministic finite automaton  $A = (Q, \Sigma, \delta, q_0, F)$ , where

```
Q = \{(p,q) \mid p \in Q_1 \text{ and } q \in Q_2\},\
q_0 = (q_{01}, q_{02}),\
F = (A_1 \times R_2) \cup (R_1 \times A_2),\
and for any (p,q) \in Q and any a \in \Sigma,
\delta((p,q), a) = \{(p',q') \in Q \mid M_1(a)[p,p'] > 0 \text{ and } M_2(a)[q,q'] > 0\}.
```

We are going to prove that the NFA  $\mathcal{A}$  accepts the language  $(L_1 \cap L_2^c) \cup (L_1^c \cap L_2)$ . If a string w of length m is accepted by the NFA  $\mathcal{A}$ , then a sequence of states  $(r_0, s_0), (r_1, s_1), \ldots, (r_m, s_m)$  exists in Q such that  $(r_0, s_0) = (q_{01}, q_{02}), (r_i, s_i) \in \delta((r_{i-1}, s_{i-1}), w[i])$  for  $i = 1, 2, \ldots, m$ , and  $(r_m, s_m) \in F$ . Thus the state  $(r_m, s_m)$  is in the set  $A_1 \times R_2$  or in the set  $R_1 \times A_2$ . In the first case, the computation  $r_0, r_1, \ldots, r_m$  is a computation of the LVFA  $\mathcal{A}_1$  on the string w ending up in an accepting state  $r_m$ , and  $s_0, s_1, \ldots, s_m$  is a computation of the LVFA  $\mathcal{A}_2$  on the string w ending up in a rejecting state  $s_m$ . Hence  $w \in L_1 \cap L_2^c$ . Similarly in the second case, the string w is in the language  $L_1^c \cap L_2$ .

Conversely, let a string w of length m is in  $L_1 \cap L_2^c$  or in  $L_1^c \cap L_2$ . In the first case, there is a computation  $q_{01}, r_1, \ldots, r_m$  of the LVFA  $\mathcal{A}_1$  on the string w ending up in an accepting state  $r_m$ , and a computation  $q_{01}, s_1, \ldots, s_m$  of the LVFA  $\mathcal{A}_2$  on the string w ending up in a rejecting state  $s_m$ . Then  $(q_{01}, q_{02}), (r_1, s_1), \ldots, (r_m, s_m)$  is an accepting computation of the NFA  $\mathcal{A}$  on the string w, *i.e.*, the string w is accepted by the NFA  $\mathcal{A}$ . The second case is similar.

Thus the NFA  $\mathcal{A}$  accepts the language  $(L_1 \cap L_2^c) \cup (L_1^c \cap L_2)$ .

We are now ready to prove the following result.

**Theorem 3.** The inequivalence problem for Las Vegas finite automata is NL-complete.

*Proof.* Consider the following nondeterministic algorithm for LVFA inequivalence problem.

```
Algorithm for LVFA inequivalence problem
```

```
Input: Two LVFAs \mathcal{A}_1=(Q_1,\Sigma,M_1,q_{01},A_1,R_1) and \mathcal{A}_2=(Q_2,\Sigma,M_2,q_{02},A_2,R_2) Task: Find whether L(\mathcal{A}_1)\neq L(\mathcal{A}_2) s:=q_{01};\, t:=q_{02};\, n:=\mid Q_1\mid\cdot\mid Q_2\mid; while n>0 do if (s\in A_1 \text{ and } t\in R_2) or (s\in R_1 \text{ and } t\in A_2) then accept and halt; n:=n-1; guess a symbol a\in\Sigma; guess a state p\in Q_1 such that M_1(a)[s,p]>0; guess a state q\in Q_2 such that M_2(a)[t,q]>0; s:=p;\, t:=q; od reject;
```

The correctness of the algorithm follows directly from Lemma 1. Only a fixed number of memory cells are used each of which is either a binary number bounded by a polynomial in the size of  $A_1$  and  $A_2$ , or can be represented by a pointer to  $A_1$  or  $A_2$ . Thus the algorithm operates in log space.

To prove NL-hardness of the problem, note that the LVFA nonemptiness problem is log space reducible to the LVFA inequivalence problem since for any LVFA  $\mathcal{A}$ , the language  $L(\mathcal{A})$  is nonempty if and only if the LVFA  $\mathcal{A}$  and the 1-state LVFA accepting the empty set are inequivalent. By Theorem 2, the LVFA nonemptiness problem is NL-complete, hence the theorem follows.

## 4. LVFA TO MINIMAL DFA PROBLEM

It is well-known that there are NFAs of n states that need  $2^n$  deterministic states [12–14]. Thus, the construction of the minimal DFA equivalent to a given NFA may require an exponential space. However, for LVFAs, the following quite surprising result was proved in [4], Theorem 4.1.

**Theorem 4** [4]. Let A be a Las Vegas automaton of n states. Then the minimal deterministic finite automaton equivalent to the LVFA A has at most  $n^2$  states.

The above result was shown in [4] by using a reduction to one-way Las Vegas communication protocols. A direct proof of the result, based on information theory, was given by Hirvensalo and Seibert in [2]. Thus, there is at most a quadratic difference between Las Vegas and deterministic finite automata. In this section, we deal with the question of how hard is it to construct the minimal DFA for a given LVFA? We will show that the construction can be done in nondeterministic polynomial time and so in polynomial space (this was the original question of Hromkovič and Schnitger).

**Lemma 2.** Testing the equivalence of a Las Vegas finite automaton and a deterministic finite automaton can be done in nondeterministic logarithmic space.

Proof. The nondeterministic log space algorithm from Theorem 3 for the inequivalence of LVFAs can be easily modified to test the inequivalence of the LVFA  $A_1$  and a DFA  $A_2 = (Q, \Sigma, \delta_2, q_{02}, A_2), R_2 = Q \setminus A_2$ : instead of guessing a state  $q \in Q_2$ , we take the state  $q = \delta_2(t, a)$ . Since NL is closed under complementation by the well-known result of Immerman and Szelepcsényi [7,17], testing the equivalence of an LVFA and a DFA is also in NL.

**Theorem 5.** The minimal deterministic finite automaton equivalent to a given Las Vegas finite automaton can be constructed in nondeterministic polynomial time.

*Proof.* The following nondeterministic algorithm first guesses a DFA of  $|Q|^2$  states, where Q is the state set of the given LVFA. Then, it tests whether the guessed DFA is equivalent to the given LVFA. If they are equivalent, then the guessed DFA is minimized:

```
Algorithm for LVFA to minimal DFA problem

Input: An LVFA \mathcal{A} = (Q, \Sigma, M, q_0, A, R)

Output: The minimal DFA \mathcal{B} equivalent to the LVFA \mathcal{A}

n := |Q|^2; S := \{1, 2, ..., n\}; s := 1; F := \emptyset;

for all q \in S do

for all a \in \Sigma do

guess a state p \in S;

\delta(q, a) := p;

od

guess a j \in \{0, 1\};

if j = 1 then F := F \cup \{q\};

od

let \mathcal{B} = (S, \Sigma, \delta, s, F);

if the LVFA \mathcal{A} and the DFA \mathcal{B} are equivalent then minimize \mathcal{B};

output (\mathcal{B});
```

By Theorem 4, the minimal DFA for the LVFA  $\mathcal{A}$  has at most  $|Q|^2$  states. By Lemma 2, testing the equivalence of the LVFA  $\mathcal{A}$  and the DFA  $\mathcal{B}$  is in NL, and so in P. Minimization of the *n*-state DFA  $\mathcal{B}$  can be done in  $\mathcal{O}(n \log n)$  time [3]. Hence the algorithm runs in polynomial time and the theorem follows.

#### 5. Conclusions

In this paper, we have obtained several results concerning the complexity of some problems for Las Vegas finite automata. We have shown that:

- the membership problem for Las Vegas finite automata is in NL;
- the nonemptiness and inequivalence problems for Las Vegas finite automata are NL-complete;

• the construction of the minimal deterministic finite automaton for a given Las Vegas finite automaton can be done in nondeterministic polynomial time

It remains open whether the membership problem is NL-hard and whether the LVFA to minimal DFA problem is NP-hard. The complexity of the minimization problem for Las Vegas finite automata [5] remains open as well.

Acknowledgements. I would like to thank Juraj Hromkovič for his comments concerning this work. I am also grateful to Professor Geffert, Alexander Szabari, and Peter Mlynarčík for our helpful discussions on the topic.

#### REFERENCES

- [1] S. Cho and D.T. Huynh, The parallel complexity of finite-state automata problems. *Inform. Comput.* **97** (1992) 1–22.
- [2] M. Hirvensalo and S. Seibert, Lower bounds for Las Vegas automata by information theory. RAIRO-Inf. Theor. Appl. 37 (2003) 39–49.
- [3] J.E. Hopcroft, An n log n algorithm for minimizing the states in a finite automaton, in The Theory of Machines and Computations, edited by Z. Kohavi. Academic Press, New York (1971) 171–179.
- [4] J. Hromkovič and G. Schnitger, On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. *Inform. Comput.* 169 (2001) 284–296.
- [5] J. Hromkovič and G. Schnitger, On the power of Las Vegas II. Two-way finite automata. Theoret. Comput. Sci. 262 (2001) 1–24.
- [6] H.B. Hunt, D.J. Rozenkrantz and T.G. Szymanski, On the equivalence, containment, and covering problems for the regular and context-free languages. J. Comput. Syst. Sci. 12 (1976) 222–268.
- [7] N. Immerman, Nondeterministic space is closed under complement. SIAM J. Comput. 17 (1988) 935–938.
- [8] T. Jiang and B. Ravikumar, A note on the space complexity of some decision problems for finite automata. *Inform. Process. Lett.* 40 (1991) 25–31.
- [9] T. Jiang and B. Ravikumar, Minimal NFA problems are hard. SIAM J. Comput. 22 (1993) 1117–1141.
- [10] N.D. Jones, Space-bounded reducibility among combinatorial problems. J. Comput. Syst. Sci. 11 (1975) 68–85.
- [11] N.D. Jones, Y.E. Lien and W.T. Laaser, New problems complete for nondeterministic log space. Math. Syst. Theory 10 (1976) 1–17.
- [12] O.B. Lupanov, A comparison of two types of finite automata. Problemy Kibernetiki 9 (1963) 321–326 (in Russian).
- [13] A.R. Meyer and M.J. Fischer, Economy of description by automata, grammars and formal systems, in *Proc. 12th Annual Symposium on Switching and Automata Theory* (1971) 188–191.
- [14] F.R. Moore, On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Trans. Comput.* 20 (1971) 1211–1214.
- [15] M. Sipser, Introduction to the Theory of Computation. PWS Publishing Company, Boston (1997).
- [16] L.J. Stockmeyer and A.R. Meyer, Word problems requiring exponential time, in Proc. 5th Annual ACM Symp. on the Theory of Computing (1973) 1–9.

510 G. JIRÁSKOVÁ

- $[17]\ {\rm R.}$ Szelepscényi, The method of forced enumeration for nondeterministic automata. ActaInform. 29 (1988) 279–284.
- [18] W.G. Tzeng, A polynomial-time algorithm for the equivalence of probabilistic automata. SIAM J. Comput. 21 (1992) 216–227.
  [19] W.G. Tzeng, On path equivalence of nondeterministic finite automata. Inform. Process.
- Lett. **58** (1996) 43–46.

 ${\bf Communicated\ by\ Juraj\ Hromkovic}.$ 

Received July 7, 2005. Accepted August 29, 2005.

To access this journal online: www.edpsciences.org