Minimal NFA Problems Are Hard

(Extended Abstract)

Tao Jiang †
Department of Computer Science and Systems
McMaster University
Hamilton, Ontario, CANADA L8S 4K1

B. Ravikumar
Department of Computer Science and Statistics
University of Rhode Island
Kingston, RI 02881, U.S.A.

Abstract

We study the complexity of the problem of converting a deterministic finite automaton (DFA) to a minimum equivalent nondeterministic finite automaton (NFA). More generally, let A $\rightarrow$ B denote the problem of converting a given FA of type A to a minimum FA of type B. We show that many of these optimal-conversion (or minimization) problems are computationally *hard*. We also study the complexity of decision problems for finite automata and present many fundamental decision problems which are computationally intractable even when the input is a DFA or a NFA with limited nondeterminism (such as unambiguous FA, or DFA's extended with one nondeterministic operation).

1. Introduction

Regular languages and their generating devices, the finite state machines, occupy a central place in theoretical computer science. Perhaps the best reason for their appeal are the elegant and diverse ways in which they can be characterized (*e.g.* in terms of finite automata, regular expressions, 2-way finite automata and others) and the nice structural properties they possess. Finite automata are easier to design and analyze than other models of computation, yet they offer many challenging problems and provide insights into some fundamental concepts in computation theory such as nondeterminism. In fact, it is a historical fact that the notion of nondeterminism was first introduced in finite automata by Rabin and Scott [Ra59].

Among the central problems in this theory are the complexity of decision problems. In a pioneering work, Stockmeyer and Meyer [St73], [Me72], and Hunt *et al.* [Hu73], [Hu74],

This work was supported in part by a grant from SERB, McMaster University, and NSERC Operating Grant OGP 0046613.

[Hu76] classified the complexity of decision problems for finite automata. Their results indicate that most of the decision problems about nondeterministic finite automata are *'hard'*. Hunt and Rosenkrantz [Hu74] have also presented 'meta-theorems' from which the *NP-hardness* and other *hardness* results for a wide class of properties of NFA can be deduced. However there are very few *hardness* results involving deterministic finite automata (DFA). Of course, many decision problems involving DFA's are not 'hard', *i.e.*, they can be solved in polynomial time and/or poly-log space. But when they are, proving *hardness* is more difficult since DFA's seem 'too weak' to encode hard problems. One of the main contributions of the present work is to establish *hardness* results involving DFA's.

Another important area of study dating back to Myhill-Nerode [My57], [Ne58] and Rabin-Scott [Ra59] is the problem of minimizing a finite automaton, or converting one type of automaton to a *minimal* automaton of another type. Some well-known results in this direction are: (i) Every NFA with n states can be converted to a DFA of size 2^n [Ra59] and in general this bound cannot be improved [Me71]. (Thus the problem of converting an NFA to a minimal equivalent DFA is provably of exponential complexity.) (ii) Every DFA with n states can be converted to a minimal equivalent DFA in $O(n \log n)$ time [Ah73]. This naturally raises the question: How about NFA minimization? It readily follows from the work of Hunt *et al.* [Hu76] and Stockmeyer and Meyer [St73] that NFA minimization is PSPACE-hard for binary alphabet and NP-hard for unary alphabet. One problem which is conspicuously absent in all of this study is the problem of converting a **DFA to a minimal equivalent NFA**. This is one of the main problems studied here. While the problem's fundamental nature is sufficient to justify its study, we think that it has practical appeal as well. Regular languages are used in many applications, and to achieve efficiency, we would like to store the regular language in a *succinct* form. Since an NFA can offer exponential saving in space, it is of interest to find minimal equivalent NFA for a given DFA. Note further that NFA is a good representation of a regular language when the principal operation performed is a membership test since membership in NFA can be tested very efficiently. One application which uses this idea is presented in [Th68].

Our study was also motivated by the following problems: (i) Minimization of *unambiguous* finite automata (UFA). A UFA is an NFA in which there is a unique accepting computation for every accepted string. UFA's were first studied by Mandel and Simon [Ma78] and Rautenauer [Ra77]. Stearns and Hunt [St85] showed that equivalence and containment problems for UFA's are decidable in polynomial time. It is of interest to know if there is a polynomial time algorithm for the (seemingly) more difficult problem of UFA minimization. (ii) Complexity of decision problems such as equivalence, containment and minimization when the NFA involved is 'structurally simple', *e.g.*, *reverse-deterministic*. (An NFA M is reverse deterministic M is obtained by 'reversing' a DFA). We show that the presence of even one nondeterministic operation can make the decision problems extremely *hard.*

This paper consists of four sections, not including this introduction. In Section 2, we introduce the technical terms and the background needed for reading the rest of the paper. In Section 3, we consider the problem of converting a given DFA to a minimal UFA and to a minimal NFA. We show in Theorem 3.1 and 3.2 that these problems (more precisely, their decision versions) are NP-complete and PSPACE-complete, respectively. In Section 4, our aim is two fold: In Section 4.1, we study the equivalence and containment problems involving DFA's augmented by just one nondeterministic operation and show that these problems are *hard.* In Section 4.2, we pursue another variation of the DFA to NFA conversion problem. We require the resulting NFA to be structurally 'simple', for example, given a DFA M_1 and an integer k, we want to determine: "Are there DFA's M_2 and M_3 such that $|M_2| + |M_3| \leq k$ and $L(M_1) = L(M_2) \cdot L(M_3)$?" In Section 5, we present some open problems arising from our work.

In this extended abstract, we omit all the proofs. A full journal version of the paper containing all the proofs can be obtained from the authors.

2. Preliminaries

We begin by introducing the technical terms and concepts from formal languages, finite automata theory and computational complexity. In this section our main goal is to introduce abbreviations (such as DFA, reduction etc.) used in this paper. We assume that the reader is familiar with fundamental notions in the above topics, and we refer the reader to a standard reference such as [Ho79] for unexplained terms.

This work is primarily concerned with two classes of finite state acceptors - DFA, the deterministic finite automata and NFA, the nondeterministic finite automata over a finite alphabet. The size of a machine M, denoted by $|M|$, is its number of states. We also study another important class - UFA, the collection of unambiguous NFA's [Ma78], [Re77], [St85]. An NFA is called unambiguous if every accepted string has a unique accepting path in the computation tree. Unambiguity has played an important role in the theory of context-free languages. Many problems about context-free languages become easier for unambiguous languages than for in the unrestricted case. For example, the universe problem† is decidable for unambiguous context-free languages, while it is undecidable for general context-free languages. Similar differences exist in sequential parsing (Earley's algorithm runs in $O(n^2)$ time while no matching upper bound is known in general), parallel parsing ($O(log\ n)$ versus $O(log^2\ n)$ time) and others. Ambiguity

The universe problem for a machine M is the question: 'Is $L(M) = \Sigma^*$?' where Σ is the alphabet over which M is defined.

plays an important role in the theory of finite state machines as well. It makes some fundamental decision problems easier. Stearns and Hunt [St85] have shown, for example, that containment and equivalence problems are decidable in polynomial time for UFA (where as the general problem is PSPACE-complete).

We use the operations on languages in the usual sense. The standard ones are: union, intersection, concatenation, Kleene star, and reversal denoted respectively by: $\cup$, $\cap$, $\cdot$, $*$ and *rev*. It is well known [Ho79] that regular languages are closed under all these operations. A DFA (or NFA or UFA) M is said to be minimum or minimal (we use these two terms interchangeably) if there is no DFA with fewer than $|M|$ states accepting $L(M)$.

The complexity theory notions used in this work are: Turing Machine (TM), NP-completeness, PSPACE-completeness and polynomial time many-one reductions, instantaneous descriptions (ID), halting TM, acceptance, time and space complexities. We refer the readers to [Ho79] for these terms. Throughout this paper, a 'reduction' means a polynomial time many-one reduction. We use the word *hard* in the technical sense that the problem is NP-hard or PSPACE-hard, or intractable in some other sense.

3. Complexity of Finding Minimal NFA

The problems we study in this section are variations of the problem of converting a given finite state acceptor to a *minimal* finite state acceptor of another type. Let A and B be two classes of finite state acceptors. For example, A may be the collection of NFA's, B the collection of DFA's. $A \rightarrow B$ denotes the problem of converting a type-A finite automaton to a minimal type-B finite automaton. It is more convenient to state this as a decision problem: The input is M, a type-A FA and an integer k (in binary). The decision question we want to answer is: "Is there a k-state type-B machine accepting $L(M)$?" We will be primarily interested in the decision version of the conversion problem. Since all the results in this section are lower bound results (NP-completeness or PSPACE-completeness) they imply the same lower bound for the optimization versions. Our study involves DFA's, NFA's and UFA's.

The two problems DFA $\rightarrow$ DFA and NFA $\rightarrow$ DFA are classical. Hopcroft's well known algorithm [Ah73] for the former problem runs in $O(n\ log\ n)$ time. The latter problem is known to have an *exponential* lower bound (on space and time), as seen from the simple fact that the minimal equivalent DFA of an n-state NFA can have 2^n states [Me71]. The decision version of the latter problem is also *hard*: It can be shown to be PSPACE-hard using the fact that the universe problem for NFA is PSPACE-hard [Me72]. In fact, the problem is NP-hard even over a one-letter alphabet [St73]. Note that these claims hold even if we fix $k = 1$. The problem NFA $\rightarrow$ UFA also requires exponential space (and thus time) since there is an unavoidable exponential blow-up when converting NFA to

UFA [St85]. The exponential blow-up holds even in the case of unary alphabet [Ra89]. The decision version of this problem is also PSPACE-hard as the argument for NFA $\rightarrow$ DFA holds in this case as well.

This leaves us with four basic problems: DFA $\rightarrow$ NFA, DFA $\rightarrow$ UFA, UFA $\rightarrow$ UFA, and UFA $\rightarrow$ DFA. We study the decision versions of these problems. The first problem is PSPACE-complete. The second and third problems are NP-complete. These results hold for any alphabet of size at least 2. The case of unary alphabet is discussed in a companion paper [Ji90]. Since UFA's can be exponentially more succinct than DFA [St85], [Ra89], the fourth problem requires exponential space and time. Unfortunately, we do not know the precise complexity of the decision version of this problem at the moment. We do not know if it is in NP. It can be shown using Lemma 3.1 stated below that it is in NP if the input k is given in unary. But we do not know if either version (unary/binary k) is NP-hard or PSPACE-hard. A special case of the unary version arises in a different context in this work and in Theorem 5.6 we present a polynomial time algorithm for this special case.

From now on, $A \rightarrow B$ denotes the decision version of the problem of converting a type-A finite automaton to a minimal type-B finite automaton. We need the following lemmas in the proofs of Theorems 3.1 and 3.2 where the claimed *hardness* results are proved. Our first two lemmas are from [St85]. They are needed to prove that DFA $\rightarrow$ UFA is in NP.

Lemma 3.1. *There is a deterministic polynomial time algorithm to decide if the two given UFA are equivalent.*

Lemma 3.2. *There is a deterministic polynomial time algorithm that, given a NFA M as input, decides if M is unambiguous.*

In the proof of Theorem 3.1, we will use a variation of the Set Basis problem, which has been shown to be NP-complete by Stockmeyer [Ga78, St76]. We call our problem the Normal Set Basis problem. The problem is stated below.

Let C and B be collections of finite sets. B is said to be a *basis* of C if for each $c \in C$, there is a subcollection of B whose union is exactly c. B is said to be a *normal basis* of C if for each $c \in C$, there is a pairwise disjoint subcollection of B whose union is exactly c. The (Normal) Set Basis problem is as follows: Given a collection C of finite sets and positive integer $k \leq |C|$, decide if C has a (normal) basis with cardinality at most k. Our proof also implies the NP-completeness of the Set Basis problem.

Lemma 3.3. *The Normal Set Basis problem is NP-complete.*

The proof of NP-hardness is by a reduction from the Vertex Cover problem [Ga78].

We now present the main results of this section.

Theorem 3.1. *DFA $\rightarrow$ UFA problem is NP-complete.*

NP-hardness is proved by a reduction from Normal Set Basis problem.

The next corollary easily follows from Lemma 3.1 and Theorem 3.1.

Corollary 3.1. *UFA $\rightarrow$ UFA problem is NP-complete.*

Our next result is proved by a reduction from the Universe Problem for Multiple DFA stated below: Given a collection of DFA's M_1, ..., M_n over a finite alphabet Σ, decide if $\bigcup_i L(M_i) = \Sigma^*$. This problem is readily seen to be PSPACE-complete by reduction from Finite State Automata Intersection problem which is known to be PSPACE-complete [Ko77]. (All we need to do is to complement the languages $L(M_i)$ by interchanging the accepting and nonaccepting states.)

Theorem 3.2. *DFA $\rightarrow$ NFA problem is PSPACE-complete.*

Our proof of Theorem 3.2 uses an alphabet of size $O(n)$ where n is the number of sets in the instance of Normal Set Basis problem. It can be modified to hold for a binary alphabet:

Corollary 3.2. *The results of Theorems 3.1 and 3.2 and Corollary 3.1 hold even when the input DFA is defined over $\Sigma = \{0, 1\}$.*

The techniques of this section can be used to show that the problems of converting a DFA to a minimal regular expression (or unambiguous regular expression) are *hard* using any standard measure for the size complexity of regular expressions. (Unambiguous regular expressions are those expressions in which every derivable string has a unique derivation.)

4. Complexity of Finding Minimal NFA's of Restricted Types

Much effort has been spent in classifying the complexity of decision problems for finite state machines. The results of Stockmeyer and Meyer [St73], Hunt *et al.* [Hu76], among others, indicate that the complexity of most of the fundamental decision problems (except membership, emptiness and finiteness) involving NFA's are *hard*. A way to understand the nature of nondeterminism is to make the NFA as simple as possible and find if the decision problems are still hard. Two well-known results of this kind are: (1) Stockmeyer and Meyer [St73]: The universe problem is coNP-complete even for unary alphabet and (2) Hunt, Rosenkrantz and Szymanski [Hu76]: The equivalence problem is coNP-complete even for NFA's with no loops. The former result combined with the existence of a normal form [Ch86] for unary NFA's implies that the universe problem is

NP-hard even for NFA's *with only one nondeterministic state.* We will present several 'hard' equivalence problems involving 'structurally' simple NFA's (such as the concatenation of two DFA's). Another collection of problems studied in this section is in keeping with the central theme of this paper, namely the complexity of finding optimal NFA's equivalent to a given DFA. Here we will require the resulting NFA to be of a special type. To give a flavor of these problems, let us consider the *Concatenation Equivalence* problem: Given three DFA's M_1, M_2 and M_3, decide if $L(M_1) \cdot L(M_2) = L(M_3)$. An analogous optimization problem (which we call *Minimum Concatenation Generation problem*) is: Given a DFA M, and an integer k, find two DFA's M_1, M_2, if possible, such that $|M_1| + |M_2| \leq k$ and $L(M) = L(M_1) \cdot L(M_2)$. It turns out that both these problems are PSPACE-complete. It follows that the presence of just one nondeterministic operation such as concatenation makes the equivalence problem extremely *hard.* Of course, one can replace concatenation by any other regularity-preserving operator $\bullet$ (unary or binary) and study the $\bullet$ *Equivalence* problem and *Minimum* $\bullet$ *Generation* problem. We study these problems for all the fundamental regularity preserving unary and binary operations.

Our problems are also related to some well-studied problems in the theory of regular languages. Let us choose $\bullet = rev$ and consider the *Minimum Reverse Generation* problem. Here we are given as input a DFA M and an integer k. We want to determine if there is a k-state DFA M' such that $L(M) = L(M')^{rev}$. This is precisely the problem of finding if the *diversity* of a DFA is upper-bounded by k. 'Diversity' is an important notion introduced recently by Rivest and Schapire [Ri87] in their study of the inference of finite automata. As another example, we note that the *Minimum Kleene Star Generation* problem closely resembles the well-known *Finite Power Property* problem [Sa81]. The $\bullet$ *Equivalence* problems were also motivated by the fact that when certain operations like concatenation or Kleene star are applied, the resulting languages require DFA's with exponentially more states than the DFA for the original language(s) [Ra89]. Thus the standard method of converting to DFA and testing equivalence will not work. We are therefore led to seek other methods to answer these decision questions or demonstrate that none is likely to exist.

We now turn to technical details. The rest of this section is divided into two subsections. In Section 4.1, we study the complexity of $\bullet$ *Equivalence* problem for $\bullet \in \{ \cup, \cap, \cdot, ^*, rev \}$. In Section 4.2, we study *Minimum* $\bullet$ *Generation* problems for the same operators.

4.1. The Equivalence Problems

In this subsection, we will study the problem of *Minimum* $\bullet$ *Equivalence* problem for $\bullet \in \{ \cup, \cap, \cdot, ^*, rev \}$. It is easy to see that this problem is decidable in polynomial time for union and intersection since we can find efficiently a 'small' DFA accepting

$L(M_1) \bullet L(M_2)$ for $\bullet \in \{\cup, \cap\}$. Thus the problem is reduced to equivalence testing of two DFA's which has an almost linear time algorithm [Ah73]. The problem is more interesting for $\bullet = rev$. It is easy to see that the minimum DFA accepting $L(M)^{rev}$ can be exponentially larger than M [Ra89] so the 'conversion method' is not efficient. Nevertheless the problem can be shown to be in P. Given DFA's M_1 and M_2, we want to determine if $L(M_1) = L(M_2)^{rev}$. We design a *NFA* M_3 from M_2 by reversing its arcs and renaming the starting and accepting states. (If M_2 has more than one accepting state, we introduce a new start state and an ϵ arc from it every accepting state of M_1 and remove ϵ-moves using the standard algorithm [Ho79].) It can be easily observed that M_3 accepts $L(M_2)^{rev}$ and M_3, although nondeterministic, is *unambiguous*. Now the polynomial time algorithm of Stearns and Hunt [St85] for equivalence testing of UFA's decides the answer to the original problem with inputs M_1 and M_3. This leaves us with two basic operations: concatenation and Kleene star. It is known that both operations can blow up the number of states in minimum DFA exponentially [Ra89] so the conversion method leads to an exponential time algorithm. We show in Theorems 4.1 and 4.2 that both the problems are PSPACE-complete. Unlike the known *hardness* results which involve NFA's or an unbounded number of DFA's (such as Kozen's result on Finite Automata Intersection problem [Ko77]), these results involve only one (or two) DFA. Alternately we can view the inputs as NFA's with a very limited form of nondeterminism (involving only one nondeterministic operator). We have the following results:

Theorem 4.1. *The Concatenation Equivalence (CE) problem is PSPACE-complete.*

Theorem 4.2. *The Kleene Star Equivalence (KSE) problem is PSPACE-complete.*

4.2 Minimization Problems

In this section we present results on the complexity of *Minimum $\bullet$ Generation* problems for all the operations considered in Section 4.1, namely union, intersection, concatenation, Kleene star and reversal. We show that the problems are NP-complete for union and intersection, while PSPACE-complete for concatenation and Kleene star. We also present a pseudo-polynomial time algorithm for reversal. The latter result implies that the diversity based representation of a regular language can be obtained from a given a DFA M in time polynomial in $n = |M|$ and D, the diversity [Ri87]. (Here D is assumed to be in unary.)

Theorem 4.3. *Minimum Union Generation (MUG) and Minimum Intersection Generation (MIG) problems are NP-complete.*

The following lemma is useful in the proofs of Theorems 4.4 and 4.5.

Lemma 4.1. *For every sufficiently large positive integer n, there is a regular language L_3 over $\{0,1\}$ such that (i) the minimum DFA accepting L_3 is of size at least n^2 but at most $O(n^2)$ and (ii) L_3 can be expressed as $L_4 \cdot L_5$ such that L_4 and L_5 can be accepted by DFA's of size at most n.*

The next two results show the intractability of Minimum Concatenation and Minimum Kleene star Generation problems.

Theorem 4.4. *The Minimum Concatenation Generation problem (MCG) is PSPACE-complete.*

Theorem 4.5. *The minimum Kleene star Generation problem is PSPACE-complete.*

We conclude Section 4.2 with the Minimum Reversal Generation (MRG) problem. The problem is to find, given a DFA M and an integer k, if there is a DFA M' with at most k states such that $L(M) = L(M')^{rev}$, or equivalently $L(M)^{rev} = L(M')$. Let M^{rev} denote the 'reverse' of M obtained using the standard algorithm (see the beginning of Section of 4.2). It is easy to see that if M_1 is deterministic then M_1^{rev} is unambiguous. Thus the above problem is a special case of the problem of converting a UFA to a minimum DFA left open in Section 3. We show that MRG problem has a *pseudo-polynomial* time algorithm, *i.e.*, it has a polynomial time algorithm if the input k is bounded by a polynomial in n, the size of M, or equivalently if k is presented in unary. Note, as remarked in Section 3, that the more general problem of finding if there is a k state DFA equivalent to a given NFA is NP-hard even if k is fixed.

The MRG problem can also be stated in terms of 'diversity' of a finite automaton as defined in [Ri87]. Let M be a DFA. Two strings x and y are defined as 'equivalent' (w.r.t. M) if for all states q, $\delta(q,x)$ is an accepting state if and only if $\delta(q,y)$ is. This equivalence relation induces a partition of Σ^* and the number of equivalence classes induced is called the *diversity* of M. If $L(M_1) = L(M_2)$ then the diversity of M_1 is equal to the diversity of $L(M_2)$ so we can define the diversity of a regular language L as the diversity of *DFA* accepting L. It has been observed by Neal Young and Dana Angluin [Sc88] that the diversity of a regular language L is the size of the minimum DFA accepting L^{rev}. Thus the MRG problem can be restated as the problem of finding, given k and a DFA M, if the diversity of $L(M)$ is bounded by k.

Theorem 4.6. *There is a pseudo-polynomial time algorithm for MRG problem.*

5. Conclusions

Complexity of decision problems for various types of language generating mechanisms has been a favorite area of research since the pioneering work of Cook [Co71] and Karp [Ka72]. Such problems have many applications including text editing, data storage and retrieval and parsing. The problem of converting one type finite state automaton to an optimal finite automaton of another type is a fundamental one. Our main contribution is to complement the classical work of [Ra59], [Me71], Me72], [St73], [Hu76] by showing that DFA $\rightarrow$ NFA and other related problems are *hard*.

Our results in Section 4 reveal that even the weakest forms of nondeterminism can render a decision problem intractable. These results substantially strengthen the known *hardness* results where the input machines are generally assumed to have unrestricted nondeterminism. Our proof techniques are new; they may find wider applications in the study of other related problems.

Our work has settled most of the fundamental questions in the proposed area of study. Yet there are some questions that are not resolved. We conclude with a list of some of them:

(i) Can the study in Section 4 be generalized to an arbitrary regularity preserving operation? Such a study should be able to state qualitatively what makes, for example, Minimum $\bullet$ Equivalence problem easy for $\bullet = rev$, but hard for $\bullet = \cdot$.

(ii) Is Minimum Reverse Generation problem NP-complete when k is presented in binary?

(iii) What is the complexity of the decision version of UFA $\rightarrow$ DFA? Is it NP-complete? PSPACE-complete?

(iv) What is the complexity of converting DFA to an *approximately* optimal NFA? More specifically, let $nsize(L)$ be the number of states in the minimal NFA accepting L and let k be a fixed integer. The problem is: Given a DFA M accepting a language L, and integer k, design an NFA accepting L with at most $(nsize(L)^k)$ states.

Acknowledgements

We are very grateful to Wing Ning Li for once again bringing the DFA $\rightarrow$ NFA problem to our attention. We would like to thank Ed McDowell for carefully reading the manuscript and suggesting improvements and for pointing out an omission in Theorem 4.1. We thank Larry Stockmeyer for sending us the technical report [St76].

References

[Ah73] Aho, A., J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, 1973, Addison-Wesley, Reading, Massachusetts.

[Ch86] Chrobak, M., 'Finite automata and unary languages', *Theoretical Computer Science* **47**, 1986, 149-158.

[Co71] Cook, S., 'Complexity of theorem proving procedures', *Proc. of 3rd Annual ACM Symp. on Theory of Computing*, 1971, 151-158.

[Ga78] Garey, M. and D.Johnson, *Computers and Intractability:A Guide to NP-Completeness*, 1978, Freeman, San Fransisco.

[Ho79] Hopcroft, J. and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, 1979, Addison-Wesley, Reading, Massachusetts.

[Hu73] Hunt, H., 'On the time and tape complexity of languages', *Proc. of 5th Annual ACM Symp. on Theory of Computing*, 1973, 10-19.

[Hu74] Hunt, H. and D. Rosenkrantz, 'Computational Parallels between regular and context-free languages', *Proc. of 6th Annual ACM Symp. on Theory of Computing*, 1974, 64-74.

[Hu76] Hunt H., D. Rosenkrantz and T. Szymanski, 'On the equivalence, containment and covering problems for the regular and context-free languages', *Jl. of Comp. and Sys. Sciences* **12**, 1976, 222-268.

[Ji90] Jiang, T., E. McDowell and B. Ravikumar, 'The structure and complexity of minimal NFA's over a unary alphabet', *University of Rhode Island Tech. Report, TR-200-90* (submitted).

[Ka72] Karp, R., 'Reducibility among combinatorial problems', in R. E. Miller and J. W. Thatcher (eds.,), *Complexity of Computer Computations*, 1972, Plenum Press, NY, 85-103.

[Ko77] Kozen, D., 'Lower bounds for natural proof systems', *Proc. of 18th Annual IEEE Symp. on FOCS*, 1977, 254-266.

[Ma78] Mandel, A. and I. Simon, 'On finite semi-groups of matrices', *Theoretical Computer Science* **5**, 1978, 183-204.

[Me71] Meyer, A., and M. Fischer, 'Economy of description by automata, grammars and formal systems', *Proc. of 12th Annual IEEE Symp. on Switching and Auotmata Theory*, 1971, IEEE Computer Society, Washington D.C., 188-191.

[Me72] Meyer, A., and L. Stockmeyer, 'The equivalence problem for regular expressions with squaring requires exponential space', *Proc. of 13th Annual IEEE Symp. on Switching and Automata Theory*, 1972, IEEE Computer Society, 125-129.

[My57] Myhill, J., 'Finite automata and the representation of events', *WADD TR-57-624*, 112-137, 1957, Wright Patterson AFB, Ohio.

[Ne58] Nerode, A., 'Linear automaton transformations', *Proc. of AMS* 9, 1958, 541-544.

[Re77] Reutenauer, C., 'Propietes arithmetiques et topologiques de series rationelles en variables noncommutatives', *These troisieme cycle*, 1977, Université de Paris VI, Paris, France.

[Ra59] Rabin, M. and D. Scott, 'Finite automata and their decision problems', *IBM Journal of Res. and Development* 3, 1959, 114-125.

[Ra89] Ravikumar, B. and O. Ibarra, 'Relating the type of ambiguity to the succinctness of their representations', *SIAM Journal on Computing* 18, 1989, 1263-1282.

[Ri87] Rivest, R. and R. Schapire, 'Diversity-based inference of finite automata', *Proc. of 28th Annual Symp. on FOCS*, 1987, 78-87.

[Sa81] Salomaa, A., *Jewels of Formal Language Theory*, 1981, Computer Science Press.

[Sc88] Schapire, R., 'Diversity based inference of finite automata', Master's thesis, MIT Laboratory for Computer Science, May 1988. *Tech. Report MIT/LCS/TR-413.*

[St85] Stearns, R., and H. Hunt, 'On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata', *SIAM Jl. on Computing* 14, 1985, 598 - 611.

[St73] Stockmeyer, L. and A. Meyer, 'Word problems requiring exponential time' (prelim. report), *Proc. of 5th Annual ACM symposium on Theory of Computing*, 1973, 1-9.

[St76] Stockmeyer, L., 'Set basis problem is NP-complete', Report No. RC-5431, 1976, IBM Research Center, Yorktown Heights, NY.

[Th68] Thompson, K., 'Regular expression searching algorithm', *Communications of ACM*, 11:6, 1968, 419-422.