SIAM J. COMPUT.
Vol. 22, No. 6, pp. 1117–1141, December 1993

MINIMAL NFA PROBLEMS ARE HARD*

TAO JIANG† AND B. RAVIKUMAR‡

Abstract. Finite automata (FA's) are of fundamental importance in theory and in applications. The following basic minimization problem is studied: Given a DFA (deterministic FA), find a *minimum* equivalent nondeterministic FA (NFA). This paper shows that the natural decision problem associated with it is PSPACE-complete. More generally, let A $\rightarrow$ B denote the problem of converting a given FA of type A to a minimum FA of type B. This paper also shows that most of these problems are computationally *hard*. Motivated by the question of how much nondeterminism suffices to make the decision problem involving an NFA computationally hard, the authors study the complexity decision problems for FA's and present several intractability results, even for cases in which the input is deterministic or nondeterministic with a very limited nondeterminism. For example, it is shown that it is PSPACE-complete to decide if $L(M_1) \cdot L(M_2) = L(M_3)$, where M_1, M_2, and M_3 are DFAs. These problems are related to some classical problems in automata theory (such as deciding whether an FA has the finite power property), as well as recent ones (such as determining the diversity of a given FA).

Key words. finite automation, minimization, NP-complete, PSPACE-complete

AMS subject classifications. 68Q15, 68Q45

1. Introduction.

Regular languages and their generating devices, the finite-state machines, occupy a central place in theoretical computer science. Among the reasons for their appeal are the elegant and diverse ways in which they can be characterized (e.g., in terms of finite automata, regular expressions, two-way finite automata) and the nice structural properties they possess. Finite automata are easier to design and analyze than are 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, the notion of nondeterminism was first introduced in automata theory by Rabin and Scott in their study of finite automata [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, Rosenkrantz, and Szymanski [Hu73], [Hu74], [Hu76] classified the complexity of a number of decision problems for finite automata. Their results indicate that these decision problems (with the exceptions of finiteness and emptiness) are computationally hard[1] when the input is a nondeterministic finite automaton (NFA). Hunt and Rosenkrantz [Hu74] also presented meta-theorems from which hardness results for a wide class of properties of NFA's can be deduced. However, very few hardness results involving deterministic finite automata (DFA's) have been shown. Of course, many decision problems involving DFA's are efficiently solvable. But when they are not efficiently solvable, proving hardness is considerably more difficult. One of the main contributions of the present work is to establish such results.

Another popular area of study dating back to Myhill [My57], Nerode [Ne58], and Rabin and Scott [Ra59] is the problem of minimizing a finite automaton or of converting one type

*Received by the editors January 11, 1991; accepted for publication (in revised form) March 22, 1992. An extended abstract of this work was presented at the 18th International Colloquium on Automata, Languages and Programming, Madrid, Spain, July 1991.

†Department of Computer Science and Systems, McMaster University, Hamilton, Ontario L8S 4K1, Canada. The research of this author was supported in part by a grant from the Science and Engineering Research Board, McMaster University, and by Natural Sciences and Engineering Research Council of Canada operating grant OGP 0046613.

‡Department of Computer Science and Statistics, University of Rhode Island, Kingston, Rhode Island 02881. The research of this author was supported in part by National Science Foundation grant CCR-9111980.

[1]In this paper a problem is computationally hard if it is NP-hard or PSPACE-hard.

of automaton to a minimum-state automaton of another type (which we call the minimum-conversion problem). Some well-known results of this kind are the following: (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 probably of exponential complexity.) (ii) Every DFA with n states can be converted to a minimal equivalent DFA in $O(n \log n)$ time [Ho71]. This naturally raises the question, How about NFA minimization? It readily follows from the work of Hunt, Rosenkrantz, and Szymanski [Hu76] and of Stockmeyer and Meyer [St73] that NFA minimization is PSPACE-hard for a binary alphabet and NP-hard for a unary alphabet. One problem that 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. Although 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 optimize the space requirements we would like to store the language in a succinct form. Since an NFA can offer exponential saving in space, it is of interest to find a minimal equivalent NFA for a given DFA. Note further that an NFA is a good representation of a regular language when the principal operation performed is a membership test. A well-known application that uses this idea is presented in [Th68]. It was suggested by a referee of this paper that an additional motivation for finding the minimal NFA equivalent to a DFA comes from computational learning theory [Na91] since in the learning theory setting the inferred machine can be an NFA. (On a related note, it is observed in [Tz89] that an equivalence algorithm for the class of probabilistic finite automata leads to an efficient learning of that class.) Unfortunately, as shown in Theorem 3.2 below, this problem is PSPACE-hard.

Our study was also motivated by a set of problems, some of which are related to the DFA $\rightarrow$ NFA conversion problem stated above:

(i) Minimization of *unambiguous* finite automata (UFA's). 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 by Reutenauer [Re77]. Stearns and Hunt [St85] showed that equivalence and containment problems are decidable in polynomial time when the inputs are UFA's. It is therefore of interest to know whether there is a polynomial-time algorithm for minimizing a UFA. We show that UFA minimization is NP-complete.

(ii) Complexity of decision problems, such as equivalence, containment, and minimization, when the NFA involved is structurally simple, e.g., reverse deterministic.[2]

Two main contributions of this work (corresponding to §§3 and 4, respectively) can be summarized as follows:

(i) We provide a complete treatment of the complexity of minimal-conversion problems involving various types of finite automata.

(ii) We prove results that substantially strengthen the previous results on the complexity of decision problems related to NFA's.

The conclusion that extremely weak forms of nondeterminism suffice to encode hard problems provides motivation for reorienting the goal towards approximate minimization problems and a careful study of heuristic methods. It should be observed that the only result that establishes the hardness of an approximation problem in automata theory is the recent result of Pitt and Warmuth [Pi89]. Our results also raise some complexity questions in classical automata theory and about some recently developed notions in finite automata. For example, by using the techniques presented in this paper it can be shown that the problem of determining the exact rank of a finite automaton is PSPACE-hard. (See Salomaa [Sa81] for a definition of the rank of a regular language.) The complexity of determining whether the rank is finite remains open. The decidability of this well-known problem is a jewel presented in [Sa81].

[2]An NFA M is reverse deterministic if the reverse of M is deterministic.

This paper consists of four sections, not including this introduction and an appendix. In §2 we introduce the technical terms and the background needed for reading the rest of the paper. In §3 we consider the problem of converting a given DFA to a minimal UFA and to a minimal NFA. We show in Theorems 3.1 and 3.2 that these problems (more precisely, their decision versions) are NP-complete and PSPACE-complete, respectively. In §4 our aim is twofold: in §4.1 we study the equivalence and containment problems involving DFA's augmented by just one nondeterministic operation (such as concatenation) and show that these problems are hard. Note that this claim is stronger than the previously known result that one nondeterministic state in the input automaton is sufficient to make a problem computationally hard. In §4.2 we pursue another variation of the DFA $\rightarrow$ NFA conversion problem in which we require that the resulting NFA be structurally simple. For illustration, a problem of this type may be stated as follows: Inputs are a DFA M_1 and an integer k. The question is, 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 §5 we present some open problems arising from our work. In the appendix we formally state the decision problems introduced and studied in this paper (as well as those used in our reductions) in a standard format.

2. Preliminaries.

We begin by introducing the technical terms and concepts from formal languages, finite automata theory, and computational complexity. Our main goal here is to introduce the necessary technical terms. We assume that the reader is familiar with fundamental notions in the above topics, and we refer the reader to standard references, such as [Ho79] and [Ga78], 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 (which is usually implicit).[3] Our definitions of these are standard; we use a 5-tuple $\langle Q, \Sigma, \delta, q_0, F\rangle$. But we will usually provide informal descriptions of machines and avoid formalism whenever possible. The size of a machine M, denoted by $|M|$, is its number of states. The conventional definition of DFA requires a DFA to be *completely specified*, i.e., a move must be defined for each pair of state and symbol. This forces us to include a dead state in most of the DFA's. (A dead state is a state from which no final state can be reached.) One can also allow a DFA to have some unspecified moves. Such a DFA is called a *partially specified* DFA. Although all of our results hold for both completely and partially specified DFA's, to simplify the presentation we will consider only partially specified DFA's in this paper. It is easy to see that all the proofs (possibly with some minor modifications) also work for completely specified DFA's. For partially specified DFA's we can further assume without loss of generality that each DFA considered consists of live states only. (A state is called live if it is not dead.) For convenience, an NFA is assumed to be ϵ-free except in the proof of Theorem 3.2, where we use ϵ-moves. (Note that the requirement that it be ϵ-free will not increase the size of a minimal NFA.) We also study a less familiar class of finite automata—UFA, the collection of unambiguous NFA's [Ma78], [Re77], [St85]. In §3 we briefly describe their significance.

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. As is customary, we drop the operator $\cdot$ except where its absence could cause discomfort in reading the expression. We also use the quotient operation (/) defined as follows: Let L be a language and x a string. Then L/x is defined as

$$L/x = \{y | y \cdot x \in L\}.$$

[3] DFA, NFA, etc., denote the collection of machines and also an individual machine. The context will easily resolve this ambiguity.

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 basic complexity theory notions used in this work are Turing machine (TM), NP-completeness, PSPACE-completeness and polynomial-time many–one reductions, instantaneous descriptions (ID's), halting TM, acceptance, and time and space complexities. We refer the readers to [Ho79] for definitions of 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, PSPACE-hard, or intractable in some other sense. The problems studied in this paper and the known hardness results used in the reductions are all listed in the appendix in a format popularized by Garey and Johnson [Ga78].

3. Complexity of finding minimal NFA.

In this section we study the minimal-conversion problems. Let A and B be two classes of finite-state acceptors. For example, A may be the collection of NFA, and B may be the collection of DFA. $A \to B$ 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 will be M, a type-A machine, 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 versions of the minimal-conversion problems. Since all the results we present are (conditional) lower-bound results (NP-completeness or PSPACE-completeness), they imply the same lower bound for the optimization versions. Our study involves three important classes of finite-state acceptors, namely, DFA, NFA, and UFA.

The two problems DFA $\to$ DFA and NFA $\to$ DFA are classical. Hopcroft's well-known algorithm [Ho71] 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 is 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 by 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 $\to$ UFA also requires exponential space (and thus time) since there is an unavoidable exponential blowup when NFA is converted to UFA [St85]. The exponential blowup holds even in the case of a unary alphabet [Ra89]. The decision version of this problem is also PSPACE-hard since the argument for NFA $\to$ DFA holds in this case as well.

This leaves us with four basic problems: DFA $\to$ NFA, DFA $\to$ UFA, UFA $\to$ UFA, and UFA $\to$ DFA. We will study the decision versions of these problems, and we will show that the first problem is PSPACE-complete and that the second and third are NP-complete. These results hold for any alphabet of size at least 2. The unary case is studied in a companion work [Ji90] (see §5 for some details regarding [Ji90]). Since UFA is exponentially more succinct than DFA [St85], the fourth problem requires exponential space and time. Unfortunately, we do not know the complexity of the decision version of this problem at the moment. (In fact, we do not know whether it is in NP or PSPACE-complete.)

From now on $A \to B$ 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 $\to$ UFA is in NP. For the sake of completeness we will present a sketch of proof of these lemmas. In fact, our proof of Lemma 3.1 is simpler than the original proof.

LEMMA 3.1 (Stearns and Hunt [St85]). *There is a deterministic polynomial-time algorithm for deciding whether the two given* UFA, *M_1 and M_2, are equivalent.*

Proof (sketch). Since a polynomial-time algorithm for containment immediately implies a polynomial-time algorithm for the equivalence problem, we concentrate on the former problem: Given M_1 and M_2, determine whether $L(M_1) \subseteq L(M_2)$. We show that the containment problem reduces to the following proper-containment problem stated as follows: Given as input two UFA's M_3, M_4 such that $L(M_3) \subseteq L(M_4)$, determine whether the containment is proper. (Reduction: Given M_1 and M_2, design a UFA M_3 for $L(M_1) \cap L(M_2)$. Note that M_3 can be obtained from M_1 and M_2 by using the standard construction [Ho79] in polynomial time. Choose $M_4 = M_1$. The reduction holds because the latter problem has a "yes" answer if and only if the former has a "no" answer.) The ideas described so far are directly from [St85]. Our simplification comes in the polynomial-time algorithm for the proper containment. Stearns and Hunt developed this algorithm by using difference equations. We use the adjacency matrix representation A and B of M_3 and M_4, respectively. It can be observed that the proper-containment problem has a "yes" answer if and only if the number of strings in $L(M_3)$ of length i is exactly equal to the number of strings of length i for each $i = 1, 2, \dots, n_1 + n_2$, where n_1 and n_2 are the number of states in M_3 and M_4. The proof now follows from the fact that the number of strings of length i can be readily obtained from the matrix A^i. A^i can be computed fast by using the standard repeated-squaring technique. □

LEMMA 3.2 (Stearns and Hunt [St85]). *There is a deterministic polynomial-time algorithm that, given an* NFA *M as input, decides whether M is unambiguous.*

Proof (sketch). Let L' be the set of strings that can be derived by at least two distinct accepting paths in M. It is easy to design an NFA M' to accept L' (from M) in polynomial time. The claim follows from the fact that M is unambiguous if and only if $L(M')$ is empty and that emptiness of an NFA can be tested in polynomial time. □

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 [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 whether C has a (normal) basis with cardinality at most k.

Since it is not obvious how the original proof of Stockmeyer in [St76] can be modified to show the NP-completeness of the normal set basis problem, here we include an independent proof of this result. Note that our proof will also show the NP-completeness of the set basis problem.

LEMMA 3.3. *The normal set basis problem is* NP-*complete.*

Proof. Clearly, the normal set basis is in NP. We show that it is NP-hard by transforming the Vertex Cover problem to it. Let $G = \langle V, E \rangle$, and k be an instance of the Vertex Cover Problem, where $V = \{v_1, v_2, \dots, v_n\}$. We design the collection C of sets as follows.

For each vertex v_i, let $c_i = \{x_i, y_i\}$, $i = 1, 2, \dots, n$. Let $\langle v_i, v_j \rangle$ be in E with $i < j$. We define

$$\begin{aligned} c_{i,j}^1 &= \{x_i, a_{i,j}, b_{i,j}\}, \\ c_{i,j}^2 &= \{y_j, b_{i,j}, d_{i,j}\}, \\ c_{i,j}^3 &= \{y_i, d_{i,j}, e_{i,j}\}, \\ c_{i,j}^4 &= \{x_j, e_{i,j}, a_{i,j}\}, \\ c_{i,j}^5 &= \{a_{i,j}, b_{i,j}, d_{i,j}, e_{i,j}\}. \end{aligned}$$

Finally, we let $C = \{c_i | 1 \le i \le n\} \cup \{c^t_{i,j} | \langle v_i, v_j\rangle \in E,\ 1 \le t \le 5\}$, and we let $s = n + 4|E| + k$. Obviously, C and s can be constructed from G and k in polynomial time. We complete the proof by showing that G has a vertex cover of size at most k if and only if C has a normal basis of cardinality at most s.

Intuitively, the idea behind the proof is as follows: To cover the set c_i, the basis B must contain either c_i or both singleton sets $\{x_i\}$ and $\{y_i\}$. Let $V_1 = \{v_i |$ both $\{x_i\}$ and $\{y_i\}$ are in $B\}$. We can show for a fixed $\langle v_i, v_j\rangle \in E$ that at least four sets (in addition to sets c_i, c_j, $\{x_i\}$, and $\{x_j\}$) are necessary to cover the five sets $c^t_{i,j}$, $t = 1, \ldots, 5$, and that four sets (in addition to sets c_i, c_j, $\{x_i\}$, and $\{x_j\}$) are sufficient if and only if at least one of v_i or v_j is in V_1. Thus if there is a vertex cover of size k, we choose the normal basis B as follows: For every v_i in the cover we include both $\{x_i\}$ and $\{y_i\}$ in B; otherwise, we include c_i in B. Conversely, if there is a normal basis of cardinality s, we will show that V_1 (defined above) can be extended to a vertex cover of size k.

Now we give the formal details. Let G have a vertex cover V_1 of size k. We will show that there is a normal basis B of size s. Define a collection of sets B as follows: For $v_i \in V_1$ include both $\{x_i\}$ and $\{y_i\}$ in B; else include $\{c_i\}$ in B. The number of sets included in B so far is $k + n$. Let $e = \langle v_i, v_j\rangle$ (where $i < j$) be an arbitrary edge in G. Since V_1 is a vertex cover, either v_i or v_j (or both) is in V_1. Assume, e.g., that v_i is in V_1. Include the sets $\{a_{i,j}, b_{i,j}\}$, $\{y_j, b_{i,j}, d_{i,j}\}$, $\{d_{i,j}, e_{i,j}\}$, and $\{x_j, d_{i,j}, e_{i,j}\}$ in B. (We omit the similar case corresponding to $v_j \in B$.) This completes the definition of B. Note that $c^t_{i,j}$ can be expressed as a union of members of B as

$$c^1_{i,j} = \{x_i\} \cup \{a_{i,j}, b_{i,j}\},$$
$$c^3_{i,j} = \{y_i\} \cup \{d_{i,j}, e_{i,j}\},$$
$$c^5_{i,j} = \{a_{i,j}, b_{i,j}\} \cup \{d_{i,j}, e_{i,j}\}$$

and that the other two sets are members of B. Since the total number of sets included in B for each edge is four, the cardinality of B is $(k + n) + 4|E| = s$. From the foregoing argument it is also obvious that B is a normal basis of C.

Conversely, suppose that there is a normal basis B of cardinality at most $s = n+4|E|+k$. We can assume without loss of generality that no proper subcollection of B is a normal basis. We show that G has a vertex cover of size at most k. Define V' $\{v_i |$ both $\{x_i\}$ and $\{y_i\}$ are in $B\}$. Let $|V'| = k'$. The number of sets in B consisting of only x_i and/or y_i is at least $n + k'$. (This can be seen from the fact that B must have the subset c_i for all i such that $v_i \notin V'$. Thus there are $n - k'$ such sets, in addition to $2k'$ singleton sets corresponding to i's such that $v_i \in V'$.) Let $E' \subseteq E$ be the set of edges covered by V', i.e., $E' = \{\langle v_i, v_j\rangle | v_i$ or v_j is in $V'\}$. The following observation can be easily shown:

Observation 3.1. For any $e \in E$ at least four sets of B (in addition to sets c_i, c_j, $\{x_i\}$, and $\{x_j\}$) are required to cover the sets $c^t_{i,j}$, $t = 1, \ldots, 5$. Further, at least five sets in addition to sets c_i, c_j, $\{x_i\}$, and $\{x_j\}$ are required to cover them if $e \notin E'$.

Now the total number of sets needed to cover C is at least

$$n + k' + 4|E'| + 5(|E| - |E'|) = n + k + 4|E| + (|E| - |E'| + k' - k).$$

Thus $|E| - |E'| \le k - k'$. Thus $|E| - |E'| + k' \le k$. We conclude the proof by showing that there is a vertex cover V of size $|E| - |E'| + k'$: Add one of the end vertices of each edge $e \in E - E'$ to V'. This vertex cover is of size $= |E| - |E'| + k' \le k$. □

We now present the main results of this section.

THEOREM 3.1. *The* DFA $\to$ UFA *problem is* NP*-complete.*

Proof. It is easy to show, by using Lemmas 3.1 and 3.2, that the problem is in NP. Let M and k be inputs. The nondeterministic algorithm will guess an NFA M' with at most k states.

It will then check, by using Lemma 3.2, that M' is unambiguous. Then, by using Lemma 3.1, it verifies that they are equal and accepts.

To prove NP-hardness we reduce the Normal Set Basis problem to our problem. Let C and s be an instance of the Normal Set Basis problem, where $C = \{c_1, c_2, \dots, c_n\}$ and $c_i = \{a^i_1, a^i_2, \dots, a^i_{n_i}\}$. Construct a DFA M as follows: M is defined over an alphabet $\Sigma = \{t \mid t \in c_i$ for some $i\} \cup \{b_i \mid i = 1, 2, \dots, n\}$. The state set of M is $Q = \{q_0, q_1, \dots, q_n, q_f\}$. The start and accepting states are q_0 and q_f, respectively. The transition function $\delta : Q \times \Sigma \to Q$ is defined as

$$\begin{aligned} &\delta(q_0, b_i) = q_i, && 1 \le i \le n, \\ &\delta(q_i, a^i_j) = q_f, && 1 \le i \le n, \quad 1 \le j \le n_i. \end{aligned}$$

Let $k = s + 2$. We claim that C has a normal basis of cardinality s if and only if $L(M)$ is accepted by a k-state UFA. Suppose C has a normal basis of cardinality at most s. Let $r_1, r_2, \dots, r_s$ be a normal basis of C. The following is a description of a UFA M' with at most k states accepting $L(M)$: Let $Q' = \{q_0, q_f, s_1, \dots, s_s\}$ be the states of M' where s_j corresponds to the basic set r_j. To describe the transition function δ' we first fix (arbitrarily) a representation of each c_i as a disjoint union of the basic sets. Say that each basic member in this representation belongs to c_i. Now we describe δ':

$$\begin{aligned} s_j \in \delta'(q_0, b_i) &\quad \text{iff } r_j \text{ belongs to } c_i, \\ \delta'(s_j, a) = q_f &\quad \text{iff } a \in r_j. \end{aligned}$$

It is easy to see that M' is a UFA and that it accepts $L(M)$.

To show the converse assume that $L(M)$ can be accepted by a UFA M' with k or fewer states. Assume M' is a minimal UFA. Let q'_0 be the starting of M'. Since $L(M)$ contains a finite set of strings of length 2, the states $Q - \{q'_0\}$ can be partitioned as Q_1 and Q_2 such that the only transitions in M' are from q'_0 to states in Q_1 and from the states of Q_1 to those in Q_2. From the minimality of M' it follows that $|Q_2| = 1$ (if it is assumed that the instance is nontrivial). For each state $q \in Q_1$ define a set $B_q = \{a \mid q_f \in \delta(q, a)\}$. It is easy to see that the collection $\{B_q\}$, $q \in Q_1$, is a normal basis of C of size at most $k - 2$. □

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

COROLLARY 3.1. *The* UFA $\to$ UFA *problem is* NP-*complete.*

We next consider the DFA $\to$ NFA problem and show that it is PSPACE-complete. We prove this result by reducing the Universe Problem for Multiple DFA to the problem DFA $\to$ NFA. The Universe Problem for Multiple DFA is the following: Given a collection of DFA $M_1, \dots, M_n$ over a finite alphabet Σ, decide whether $\bigcup_i L(M_i) = \Sigma^*$. This problem is readily seen to be PSPACE-complete by reduction from the Finite-State Automata Intersection Problem, which was shown by Kozen [Ko77] to be PSPACE-complete. (All we need to do is to complement the languages $L(M_i)$ by interchanging the accepting and nonaccepting states.)

THEOREM 3.2. *The* DFA $\to$ NFA *problem is* PSPACE-*complete.*

Proof. This problem is in PSPACE since the equivalence problem for NFA's is in PSPACE. We show that it is PSPACE-hard by reducing the universe problem for multiple DFA to our problem. Let DFA $M_1, \dots, M_n$ (over an alphabet Σ) form an instance of the universe problem. (The answer to this instance is "yes" if $\bigcup_i L(M_i) = \Sigma^*$ and is "no" otherwise.) We want to transform this to a DFA M and an integer k such that the answer to the universe problem is "yes" if and only if some NFA with k or fewer states can accept $L = L(M)$. Intuitively, the idea is as follows. We can assume that M_i are minimal. The alphabet over which L is defined includes, in addition to the symbols of Σ, additional symbols $b_{i,j}$, corresponding to state $q_{i,j}$ of M_i, and a_i, corresponding to the start state of M_i. M is chosen so as to accept the marked versions of the strings accepted by M_i's, where the marks correspond to states of M_i's. This ensures that any NFA accepting L must essentially possess all the states of M_i's. In addition, L includes Σ^* and some additional strings used as enforcers. The enforcers will be chosen in such a way that an NFA N accepting L will require at least k states and at most $k+1$ states, where $k = 4 + \sum_i |M_i|$. Since all states of M_i's are required to be present in N, if $\bigcup_i L(M_i) = \Sigma^*$, then with no additional state, N can accept all strings in Σ^*. The enforcers will make sure that neither the states of M_i's nor the states needed to accept the enforcer strings can be used for accepting Σ^*. Thus at least one more state is needed to accept L.

We now present a formal proof. We assume with no loss of generality the following properties of M_i: (i) M_i's are minimal, (ii) for each i $L(M_i)$ is neither empty nor equals Σ^*, and (iii) $\Sigma \cup \{\epsilon\}$ is contained in $\bigcup_i L(M_i)$. Let t_i be the number of states in M_i, let $Q_i = \{q_{i,1}, \dots, q_{i,t_i}\}$ be the state set of M_i, let $q_{i,1}$ be the start state, and let $F_i \subseteq Q_i$ be the accepting states of M_i. For a state $q_{i,j}$ let $P(i,j)$ be defined as the set of strings accepted by M_i if F_i were redefined to be the singleton $\{q_{i,j}\}$. More formally,

$$P(i,j) = \{w | \delta(q_{i,1}, w) = q_{i,j}\}.$$

Clearly, $P(i,j) \cap P(i,k) = \emptyset$ for $j \neq k$.

For each i, $1 \le i \le n$, let a_i be a new symbol, and for each i, j, $1 \le i \le n$, $1 \le j \le t_i$, let $b_{i,j}$ be a new symbol. Let $B = \{b_{i,j} | 1 \le i \le n,\ 1 \le j \le t_i\}$. Let c, d, and f be three additional new symbols. For each i define a language $P(i)$ as

$$P(i) = a_i \bigcup_{j=1}^{t_i} [P(i,j) b_{i,j}].$$

Also define

$$Q(i) = \{b_{i,1}\} \cup \{x b_{i,j} | x \in \Sigma, \delta(q_{i,1}, x) = q_{i,j}\}$$

and

$$R = (\{c\} \cup \Sigma)(\{d\} \cup \Sigma)\Sigma^*(\{f\} \cup B).$$

Finally, let

$$L = \bigcup_i [P(i) \cup a_i L(M_i) \cup Q(i)] \cup \Sigma^* \cup R,$$

and let $k = \Sigma_i t_i + 4$. It is easy to show that a DFA accepting L can be constructed in polynomial time, given M_i. To conclude the proof of Theorem 3.2 it suffices to prove Claim 3.2 below which asserts that the reduction is correct. Before we prove this, we need some observations concerning the languages defined above. In particular, we will establish a lower bound of k and upper bound of $k+1$ on the size of a minimal NFA accepting L (irrespective of whether the answer to the original instance is "yes" or "no"). With this intention we describe (informally) an NFA N' with $k+1$ states accepting L. N' is obtained by taking the machines M_i's and adding five more states q_0, p_1, p_2, p_3, and f. The start state is q_0, and the accepting states of N' are the accepting states of M_i's along with p_1 and f. The transitions include all the original transitions of M_i's, with the following additions: $q_{i,1} \in \delta'(q_0, \epsilon)$, and $q_{i,1} \in \delta'(q_0, a_i)$, $\delta'(q_0, a) \in f$ for all $a \in \Sigma$, $p_1 \in \delta'(q_{i,j}, b_{i,j})$, $p_2 \in \delta'(q_0, c)$, $p_2 \in \delta'(q_0, a)$ for all $a \in \Sigma$, $p_3 \in \delta'(p_2, d)$, $p_3 \in \delta'(p_2, a)$ for all $a \in \Sigma$, $p_3 \in \delta'(p_3, a)$ for all $a \in \Sigma$, and, finally, $p_1 \in \delta'(p_3, f)$, $p_1 \in \delta'(p_3, b)$ for all $b \in B$. Figure 3.1 presents this NFA. Please note that

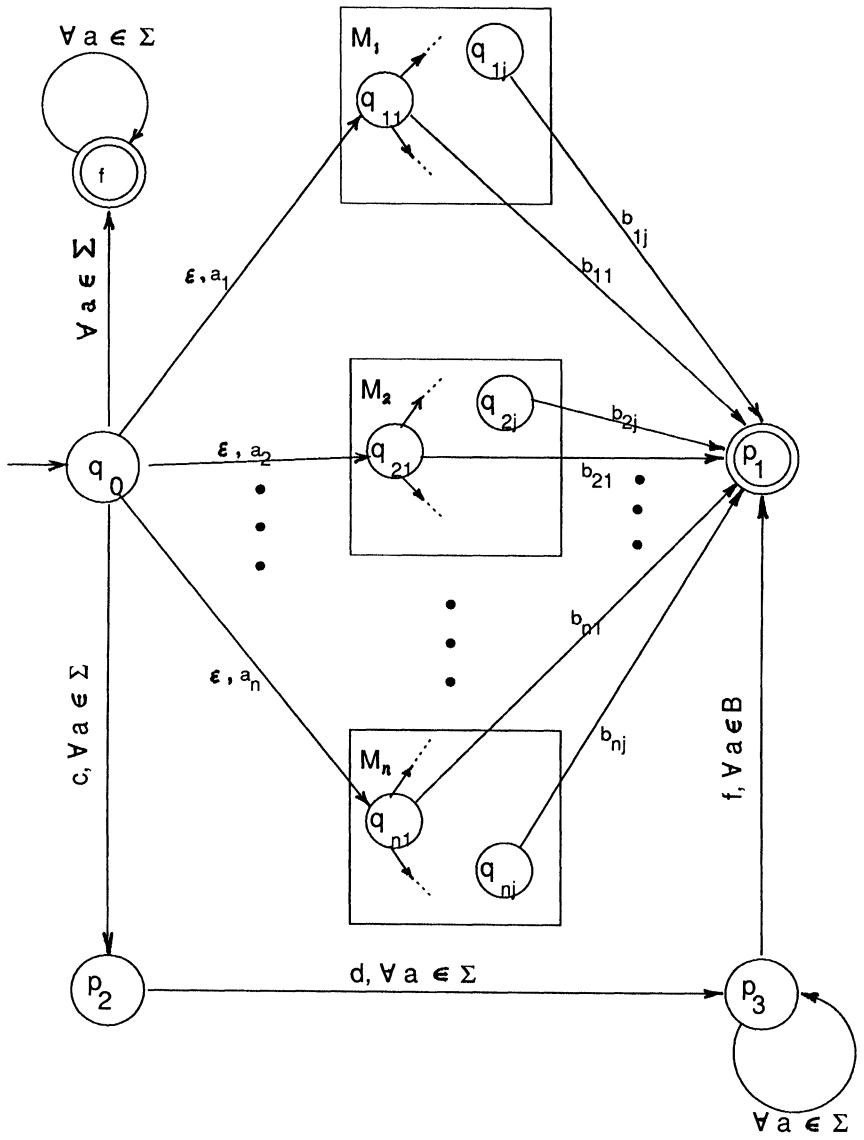

FIG. 1. NFA *in the proof of Theorem* 3.2. (*N.B. M_i's are given* DFA'*s, with all the arcs and only the additional edges connecting M_i's to the other states are shown.*)

this figure omits most of the details of the internal structure of M_i's and concentrates only on the external connections, connecting M_i's to the remaining five states.

Thus if N is a minimal NFA accepting L, $|N| \leq k+1$. Let Q_N be the set of states of N, let F_N be the accepting states of N, and let q_0 be the start state. Define $S(i, j)$ as the set of states p reachable following a string in $a_i P(i, j)$ such that some final state is reachable from p via an arc labeled $b_{i,j}$. More succinctly,

$$S(i, j) = \{q \in Q_N | \delta(q_0, w) = q \text{ for some } w \in a_i P(i, j) \text{ and } \delta(q, b_{i,j}) \cap F_N \neq \emptyset\}.$$

Clearly, $S(i, j)$ is nonempty for all pairs (i, j). We now prove the following claim:

CLAIM 3.1. *For $(i, j) \neq (i', j')$, $S(i, j) \cap S(i', j') = \emptyset$.*

Proof. Suppose not. Then there is a $u \in S(i, j) \cap S(i', j')$. We consider two cases.

Case 1: $i \neq i'$. Clearly, there exist strings x and y such that $u \in \delta(q_0, a_i x)$, $u \in \delta(q_0, a_{i'} y)$, and $\delta(u, b_{i,j}) \cap F \neq \emptyset$. This implies that $a_{i'} y b_{i,j} \in L$, a contradiction.

Case 2: $i = i'$. Clearly, $j \neq j'$. Let w be a string in $P(i, j)$ such that $u \in \delta(q_0, a_i w)$. Note that $w \notin P(i, j')$ since $P(i, j) \cap P(i, j') = \emptyset$ for $j \neq j'$. It follows that $a_i w b_{i,j'} \in L$, a contradiction. This concludes the proof of Claim 3.1. □

From Claim 3.1 it follows that N must have at least $\sum_{i=1}^{n} t_i = k - 4$ states. Also, in the above proof our argument in Case 1 implies that each state in $S(i, j)$ is not reachable from any state in $S(i', j')$ if $i \neq i'$. Our next step is to show that N needs four more states in addition to the $k - 4$ established above. The need for these additional four states is established in (i)–(iv) below:

(i) Clearly, the start state q_0 is not in any $S(i, j)$.

(ii) Consider the set of states Q_c reachable from q_0 on c. Clearly, Q_c is not empty and is different from each state in $S(i, j)$.

(iii) The same argument is used for Q_{cd} (the set of states reachable following cd from q_0). Note also that $Q_{cd} - Q_c \neq \emptyset$.

(iv) Consider the set of states with an incoming arc labeled $b_{i,j}$ for some (i, j). This set is nonempty and cannot overlap any of the states described above. Denote this set by Q_f.

Summarizing what we have shown so far, a minimal NFA accepting L will have at least k states and at most $k + 1$ states.

CLAIM 3.2. *$\bigcup L(M_i) = \Sigma^*$ if and only if L can be accepted by an* NFA *with at most k states.*

Proof. (only if) We exhibit a k-state NFA accepting L: Simply remove the state f (and all the transitions associated with it) from the NFA of Fig. 3.1. The resulting NFA is easily seen to accept L.

(if) Suppose that a minimal NFA N accepting L has k states. Define $S(i, j)$, q_0, Q_c, Q_{cd}, and Q_f as defined above for this N. It is easily seen that the state set of N is $\bigcup_{i,j} S(i, j) \cup \{q_0\} \cup Q_c \cup Q_{cd} \cup Q_f$. Also, $|S(i, j)| = 1$ for all i and j, $|Q_c| = 1$, $|Q_{cd}| = 1$, and $|Q_f| = 1$. This is true because our lower bound of k in Claim 3.1 counted only one for each $S(i, j)$ and each of Q_c, Q_{cd}, and Q_f. By the definition of R, Q_c and Q_{cd} cannot lead to a final state following any string in Σ^*. We also know from the argument presented above that a string in Σ^* can be accepted only by using some states in $\bigcup_{i,j} S(i, j) \cup \{q_0\} \cup Q_f$.

The definitions of $Q(i)$'s and R restrict the states in $\bigcup_{i,j} S(i, j)$ that N can move to in the first step on a symbol in $\Sigma \cup \{\epsilon\}$. Clearly, in the first step N can only move to $S(i, 1)$ for some i on ϵ. On a symbol $x \in \Sigma$, N can only move to some $S(i, j)$, where $q_{i,j} = \delta(q_{i,1}, x)$.

Let L_{0f} be the set of strings in Σ^* accepted by using states q_0 and Q_f only. Since N cannot move to q_0 (Q_f) from q_0 (Q_f, respectively) on any symbol,

$$L_{0f} \subseteq \Sigma \cup \{\epsilon\}.$$

For each i let L_i be the set of strings in Σ^* accepted by using states in $\{q_0\} \cup \bigcup_j S(i, j)$. For each $S(i, j)$ let $L(i, j)$ be the subset of Σ^* accepted if $S(i, j)$ is considered the start state. Then for each i

$$L_i = \bigcup \{xL(i, j) | x \in \Sigma \cup \{\epsilon\}, N \text{ can move to } S(i, j) \text{ on symbol } x\}.$$

The definition of $S(i, j)$ means that if N can move to $S(i, j)$ from q_0 on $a_i x$, then $x \in P(i, j)$. Thus

$$\begin{aligned} a_i L(M_i) &= a_i \bigcup_j P(i, j) L(i, j) \\ &\supseteq a_i L_i. \end{aligned}$$

Thus $L_i \subseteq L(M_i)$. Hence

$$\begin{aligned} L(N) \cap \Sigma^* &= \bigcup_i L_i \cup L_{0f} \\ &\subseteq \bigcup_i L(M_i) \cup L_{0f}. \end{aligned}$$

Since $\Sigma^* \subseteq L = L(N)$ and $L_{0f} \subseteq \Sigma \cup \{\epsilon\} \subseteq \bigcup_i L(M_i)$,

$$\begin{aligned} L(N) \cap \Sigma^* &= \Sigma^* \\ &\subseteq \bigcup_i L(M_i) \cup L_{0f} \\ &\subseteq \bigcup_i L(M_i). \end{aligned}$$

That is, $\bigcup L(M_i) = \Sigma^*$. This concludes the proof of Claim 3.2 and Theorem 3.2. □

It is interesting to observe that the results of Theorems 3.1 and 3.2 and Corollary 3.1 hold even when restricted to a binary alphabet. We sketch a proof of this result:

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* $\Sigma = \{0, 1\}$.

Proof (sketch). The essential idea is to use a binary encoding in the above proofs. This is very easy to do in Theorem 3.1 and Corollary 3.1. So we consider Theorem 3.2. Let the alphabet over which L of Theorem 3.2 is defined be Σ' and $m = |\Sigma'|$. For convenience, relabel the symbols of Σ' as $\{a_1, \dots, a_m\}$. We encode a_i as $0^i 1$. Formally, let h be the homomorphism $h(a_i) = 0^i 1$. To complete the reduction we need to introduce some additional enforcers. The following motivates the need for the enforcers. Consider a minimum NFA N accepting L, and let $q_i \to q_j$ be a transition on symbol a_k in N. This transition can be realized in N', the corresponding minimum NFA over the binary alphabet, by a chain of states $q_i, p_{i,1}, p_{i,2}, \dots, p_{i_k}, q_j$ as

$$q_i \xrightarrow{0} p_{i,1} \xrightarrow{0} p_{i,2} \xrightarrow{0} \cdots \xrightarrow{0} p_{i,k} \xrightarrow{1} q_j.$$

We want the new states $p_{i,1}, p_{i,2}, \dots, p_{i,k}$ to be private states of q_i, i.e., they should not be shared by any other states of N. This can be achieved by introducing additional strings in L'. For each i let $P(i)$ denote the language accepted N if q_i were considered to be the only accepting state. Define

$$L' = h(L) \cup_i (h(P(i)) 0^m 1^{i+1}).$$

Then each state q_i is forced to have m private states $p_{i,1}, p_{i,m}, \dots, p_{i,m}$ in N'. The following claim, which we state without proof, completes the proof of the corollary:

CLAIM 3.3. $\bigcup_i h(L(M_i)) = h(\Sigma^*)$ *if and only if* L' *can be accepted by an* NFA *with at most* $(m + 1)(|N| + 1)$ *states.* □

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, Rosenkrantz, and Szymanski [Hu76], and 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 determine whether the decision problems are still hard. Two well-known results of this kind are that of Stockmeyer and Meyer [St73], who showed that the universe problem is coNP-complete even for a unary alphabet, and that of Hunt, Rosenkrantz, and Szymanski [Hu76], who showed that the equivalence problem is coNP-complete even for NFA's with no loops. The former result, combined with the existence of the normal form [Ch86] for unary NFA's (see §1), implies that the universe problem is hard even for NFA's with only *one* nondeterministic state. We will present several interesting hardness results on equivalence, containment, etc., in which the NFA involved will be structurally very simple (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 the flavor of these problems, let us consider the Concatenation Equivalence problem: Given three DFA's M_1, M_2, and M_3, decide whether $L(M_1) \cdot L(M_2) = L(M_3)$. An analogous optimization problem (which we call the *Minimum Concatenation Generation problem*) is the following: 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 of 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 Op (unary or binary) and study the Op *Equivalence problem* and the *Minimum* Op *Generation problem.* We study these problems for all the fundamental regularity-preserving unary and binary operations.

The Op Equivalence problems were motivated by the fact that when certain operations, such as concatenation or Kleene star, are applied, the resulting languages require DFA's with exponentially more states than those of 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.

It should be remarked that the results presented in this section are not related to the well-developed algebraic decomposition theory of which the central result is the Krohn–Rhodes decomposition theorem [Gi68], [Ha66]. The fundamental difference between our results and the classical decomposition theory is that in the latter theory the FA is required to be decomposed into structurally simple automata (such as reset automata and permutation automata), whereas we want the constituents to be simple in the sense of being small in size.

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

4.1. Equivalence problems. In this subsection we will study the Minimum Op Equivalence problem for Op $\in \{\cup, \cap, \cdot, *, \text{rev}\}$. It is easy to see that this problem is decidable in polynomial time for union and intersection since we can efficiently find a small DFA accepting $L(M_1)$Op $L(M_2)$ for Op $\in \{\cup, \cap\}$. Thus the problem is reduced to equivalence testing of two DFA's, which has an almost linear time algorithm [Ho79]. The problem is more interesting for Op = rev. It is easy to see that the minimum DFA accepting $L(M)^{\text{rev}}$ can be exponentially larger than M [Ra89], and 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 whether

$L(M_1) = L(M_2)^{\text{rev}}$. We design an 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 to every accepting state of M_1, and we remove ϵ-moves by using the standard algorithm [Ho79].) It can be easily seen that M_3 accepts $L(M_2)^{\text{rev}}$ and that 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], and so the conversion method leads to an exponential-time algorithm. We show in Theorems 4.1 and 4.2 below that both the problems are PSPACE-complete. These results are surprising and differ from the known results in that all known hard decision problems involve NFA's or an unbounded number of DFA's (e.g., Kozen's result on the Finite Automata Intersection Problem [Ko77]). We have the following theorem:

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

Proof. CE is easily seen to be in PSPACE. In what follows we will show its PSPACE-hardness. The proof is by reduction from the Linear-Space Acceptance problem [Ga78]. Let M be a fixed (one-tape) deterministic linear-bounded automaton (Turing machine), and let w be an input to M. The question is, "Does M accept w?" We show how to transform (in polynomial time) this instance to three DFA's M_1, M_2, and M_3 such that M does not accept w if and only if $L(M_1) \cdot L(M_2) = L(M_3)$. Let $|w| = n - 1$. Let Γ be the work-tape alphabet, let Q be the set of states, and let F be the set of accepting states of M. Let ¢ and \$ be the left and right end markers on the tape. Let # be a special symbol in Γ. We assume without loss of generality that in any accepting computation M writes a # on every tape square (including the two end squares) and enters an accepting state. Assume further that M makes at least two moves on every input. Let $S = \Gamma \cup Q$.

Informally, the idea behind the proof is as follows. Let INV_w^M be the set of invalid computations of M on input w. Thus INV_w^M is either S^* (if $w \notin L(M)$) or S^* with the exclusion of *exactly* one string (namely, the valid computation of M on w, if $w \in L(M)$). We show that R defined as $R = (\text{INV}_w^M)^{\text{rev}}$, the reverse of the set of invalid computations, can be expressed as $R = L_1 \cdot L_2$ for two regular languages L_1 and L_2, both of which can be accepted by "small" DFA's. Let us describe L_1 and L_2 informally. Note that the reverse of the valid computation of M on input w is of the form $w' = \text{ID}_m^{\text{rev}}\text{ID}_{m-1}^{\text{rev}} \cdots \text{ID}_0^{\text{rev}}$, where $|\text{ID}_i| = n$ for each i, the start and the final ID are correct, and the successive ID's in the list yield the preceding ID. Thus for a string w' to be in R one of the following conditions should hold: (i) the start of accepting ID is bad, or (ii) it is not true that $\text{ID}_j \vdash \text{ID}_{j+1}$ for some j. Since condition (i) can be easily handled, let us concentrate on (ii) in this informal outline. Suppose that $\neg(\text{ID}_j \vdash \text{ID}_{j+1})$ and that the position of mismatch is t. Let $w' = x_1 \cdot x_2$, where $|x_1| = n - t$. (We set this length to $n - t$ rather than to t since we consider the reverse string.) Now the position of mismatch occurs in x_2 at a position that is a multiple of n. Thus a small DFA that counts modulo n can locate the mismatch by using a small buffer (of constant number of bits). The formal details are slightly more complicated in that the compatibility checking may require matching three symbols of ID_i with three symbols of ID_{i+1} (when a state symbol is involved in the mismatch). We should further handle the starting and accepting ID's. Now we present the formal details.

For strings $a_1a_2a_3$ and $b_1b_2b_3$ (where a_i, $b_i \in S$) we say that $a_1a_2a_3 \Rightarrow b_1b_2b_3$ if (i) $a_1, a_2, a_3 \in S - Q$ and $a_2 = b_2$ or (ii) $a_2 \in Q$ and $a_3a_2a_1 \vdash b_3b_2b_1$. Thus $\Rightarrow$ is simply an extension of $\vdash$ for strings of length three to include the identity relation. We now define language L_1 as

$$L_1 = \{x | x \in S^*, |x| \leq n - 3\}.$$

L_2 is defined as the set of strings x over S satisfying one of the following properties:

(i) The leading three symbols are not all in $\{\#\} \cup F$.

(ii) ID_0 is not a suffix of x.

(iii) Let $x = x_0 x_1 \cdots x_m$, where each $x_i \in S$. There exists a t such that $0 \leq tn \leq m - n - 2$ and it is not true that $x_{(t+1)n} x_{(t+1)n+1} x_{(t+1)n+2} \Rightarrow x_{tn} x_{tn+1} x_{tn+2}$.

Note that there are DFA's with $O(n)$ and $O(n^2)$ states recognizing L_1 and L_2, respectively, and that these DFA's can be designed in polynomial time. The claim follows from the fact that $L(M_1) \cdot L(M_2) = S^*$ if and only if M does not accept w. □

Observe that what we have shown is stronger than the claim of Theorem 4.1 in that the proof holds when M_3 accepts Σ^*. It is easy to modify the above proof to show that the following problem is PSPACE-complete: Inputs are DFA's M_1 and M_2 over an alphabet Σ such that $\epsilon \notin L(M_1)$ and $\epsilon \notin L(M_2)$. The question is, "Is $L(M_1) \cdot L(M_2) = L_0 = \{x | x \in \Sigma^*, |x| \geq 2\}$?" This stronger result is needed to prove the next result. We use the name SCE to denote this problem.

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

Proof. This problem is easily seen to be in PSPACE. To prove its PSPACE-hardness we reduce SCE to KSE. SCE's instance is specified by two DFA's M_1 and M_2 such that $\epsilon \notin L(M_1)$ and $\epsilon \notin L(M_2)$. The question to be answered is whether $L(M_1) \cdot L(M_2) = L_0 = \{x | x \in \Sigma^*, |x| \geq 2\}$. We will transform this (in polynomial time) to an instance of KSE. Let the input size of the instance of SCE be $n = |M_1| + |M_2|$. Let # be a new symbol not in Σ, and let $\Gamma = \Sigma \cup \{\#\}$.

Define the following languages:

$$\begin{aligned}
L_1 &= \#L(M_1), \\
L_2 &= L(M_2)\#, \\
L_3 &= (\Sigma^+\#\Sigma^*\#\Sigma^*) \cup (\Sigma^*\#\Sigma^*\#\Sigma^+), \\
L_4 &= \Gamma^*\#\Gamma^*\#\Gamma^*\#\Gamma^*, \\
L_5 &= L_1 \cup L_2 \cup L_3 \cup L_4, \\
L_6 &= \{\epsilon\} \cup L_1 \cup L_2 \cup L_3 \cup L_4 \cup (\#L_0\#).
\end{aligned}$$

We now make the following claims:

(i) L_5 and L_6 can be accepted by DFA's M_3 and M_4, both of size $O(n)$, and these DFA's can be constructed from M_1 and M_2 in time linear in n.

(ii) $L_5^* \subseteq L_6$ and $L_6 - L_5^* = (\#L_0\#) - (L_1 L_2)$.

The proof of (i) is obvious. For the proof of (ii) let $w \in L_5^*$. We will show that $w \in L_6$. We will consider mutually exclusive and collectively exhaustive possibilities for w.

Case 1: w does not contain any #. Clearly, $w = \epsilon \in L_6$.

Case 2: w has exactly one occurrence of #. In this case $w \in L_1 \cup L_2$, and so $w \in L_6$.

Case 3: w has at least three occurrences of #. Then $w \in L_4$, and so $w \in L_6$.

Case 4: w has exactly two occurrences of #, and w either begins or ends with a symbol in Σ. In this case $w \in L_3$, and so $w \in L_6$.

Case 5: w has exactly two occurrences of #, and w starts and ends with #. Note that this case occurs only when $w \in L_1 \cdot L_2$. (This is why we require that neither M_1 nor M_2 accept ϵ.) Thus $w \in L_1 \cdot L_2 \subseteq \#L_0\# \subseteq L_6$.

This argument further implies that $L_6 - L_5^*$ consists of exactly the set of strings of the form $\#z\#$, where $z \notin L_1 \cdot L_2$. This proves (ii).

The reduction is complete with M_3 and M_4 forming the instance of KSE. We conclude the proof by observing that $L(M_1) \cdot L(M_2) = L_0$ if and only if $L_5^* = L_6$. □

We briefly remark on the complexity of containment problems. In our setting the containment questions are of two types: Is $L(M_1)\text{Op}\, L(M_2) \subseteq L(M_3)$? and Is $L(M_3) \subseteq L(M_1)\text{Op}\, L(M_2)$? for input DFA's M_1, M_2, and M_3. Both problems are easily seen to be solvable in polynomial time for Op = union, intersection, or reversal. The last result follows from the result of Stearns and Hunt [St85] that containment is decidable in polynomial time for UFA's. Recall that containment problems involving (general) NFA's are hard [Hu76] since the universe problem is essentially the containment question "Is $\Sigma^* \subseteq L(M)$?", given M as input. The same argument carries over to concatenation since M_3 actually accepts Σ^* in Theorem 4.1. Thus the following problem is PSPACE-complete: Decide whether $L(M_3) \subseteq L(M_1) \cdot L(M_2)$ for given DFA's M_1, M_2, and M_3. However, the inclusion in the other direction is easily seen to be decidable in P:

RESULT 4.1. *There is a polynomial-time algorithm for deciding whether* $L(M_1) \cdot L(M_2) \subseteq L(M_3)$, *given* DFA'*s* M_1, M_2, M_3 *as inputs.*

Proof. Note that the above containment is equivalent to $L(M_1) \cdot L(M_2) \cap \overline{L(M_3)} = \emptyset$. We can efficiently design NFA's accepting $L(M_1) \cdot L(M_2)$ and $\overline{L(M_3)}$. We can design an NFA (whose size is the product of the sizes of the two NFA's) accepting the intersection of these languages. Finally, we can test the emptiness of the language accepted by an NFA by using the reachability algorithm. □

Similarly we can show the following:

RESULT 4.2. *There is a polynomial-time algorithm for deciding whether* $L(M_1)^* \subseteq L(M_2)$, *given* DFA'*s* M_1 *and* M_2 *as inputs.*

Result 4.3 below follows from Result 4.2.

RESULT 4.3. *The following problem is* PSPACE-*complete*:

INSTANCE: Two DFA's M_1, M_2.

QUESTION: Is $L(M_2) \subseteq L(M_1)^*$?

Proof. If there is a polynomial-time algorithm for this problem, then, in conjunction with Result 4.2, it will yield a polynomial-time algorithm for KSE that will imply that PSPACE = P. □

4.2. Minimization problems. In this section we present results on the complexity of Minimum Op Generation problems for all of the operations considered in §4.1, namely, union, intersection, concatenation, Kleene star, and reversal. We show that the problems are NP-complete for union and intersection and that they are 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 DFA M in time polynomial in $n = |M|$ and D, the diversity [Ri87]. (Here D is assumed to be in unary.)

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

Proof. We will prove the result only for MUG since the NP-completeness of MIG readily follows from that of MUG by complementation. MUG is clearly in NP. We prove its NP-hardness by reduction from the Minimum Inferred DFA problem [Go78]. Let Σ be a finite alphabet, let $S, T \in \Sigma^*$ be two finite sets of strings (called the positive and negative samples, respectively), and k an integer form an instance of the Minimum Inferred DFA problem. The question is, "Is there a DFA with at most k states such that M accepts *every* string in S and none in T?" (We do not care about the behavior of M on other strings.) We say that such an

M is *consistent* with S and T. We assume without loss of generality that $S \cap T = \emptyset$. Let # be a new symbol not in Σ. For a regular language L let *size*(L) denote the size of the minimum DFA accepting L. Let $m = k +$ size$(\overline{T} \cap \overline{S})$.

Define the following languages:

$$\begin{aligned} L_1 &= \overline{T}, \\ L_2 &= \overline{T} \cap \overline{S}, \\ L_3 &= (L_2\#^m)^+ \quad \text{(recall the definition of } m \text{ above)}, \\ L_4 &= L_1\#^m, \\ L_5 &= L_3 \cup L_4. \end{aligned}$$

We prove the following claims:

CLAIM 4.1. *Let L be a language. Then*

$$\text{size}(L\#^m) = \text{size}((L\#^m)^+) = m + \text{size}(L).$$

Proof. Let M be the minimum DFA accepting L. A DFA M' accepting $L\#^m$ can be designed by adding a #-tail of length m to M and by letting $\delta(f, \#) = p_0$ for every final state f of M, where p_0 is the first state in the #-tail. Clearly, $L(M') = (L\#^m)$, so size$(L\#^m) \leq$ size$(L) + m$. To prove the reverse inequality let M' be the minimum DFA accepting $L\#^m$. We can obtain a DFA M from M' by removing all states having an incoming #-arc. We would have removed exactly m states in this process, and so $|M| = |M'| - m$ and the claim follows. We can also show that size$((L\#^m)^+) =$ size$(L\#^m)$. This concludes the proof of Claim 4.1. □

CLAIM 4.2. $L_1\#^m = (L_2 \cup L(M'))\#^m$ *for any* DFA *M' consistent with S and T.*

Proof. Since $S \cap T = \emptyset$, $S \subseteq \overline{T}$. Thus

$$\begin{aligned} L_1 &= \overline{T} \\ &= (\overline{T} \cap \overline{S}) \cup (\overline{T} \cap S) \\ &= \overline{T} \cap \overline{S} \cup S \\ &= L_2 \cup S. \end{aligned}$$

Observe that $S \subseteq L(M') \subseteq \overline{T}$ and that $L_2 \cup L(M') \subseteq L_1$. We have

$$\begin{aligned} L_1\#^m &= (L_2 \cup S)\#^m \\ &\subseteq (L_2 \cup L(M'))\#^m \\ &\subseteq L_1\#^m \qquad \text{(since } L_2 \cup L(M') \subseteq L_1). \end{aligned}$$

This concludes the proof of Claim 4.2. □

Now we present the reduction. Let M be the minimum DFA accepting L_5. The instance of MUG to which S, T, k has been transformed is M, $3m$. Let $n = |S| + |T| + k$ be the size of the instance of the Minimum Inferred DFA problem. It is easy to see that M can be constructed from S, T, and k in time bounded by a polynomial in n. The correctness of the reduction is stated as the next claim.

CLAIM 4.3. *There is a k-state* DFA *consistent with S and T if and only if $L(M) = L_5$ is the union of $L(M_1)$ and $L(M_2)$ for two* DFA's *M_1 and M_2 such that $|M_1| + |M_2| \leq 3m$.*

Proof. (only if) Suppose that there is a k-state DFA M' consistent with S and T. Let M_1 be the minimum DFA accepting L_3, and let M_2 be the minimum DFA accepting $L(M')\#^m$. We show that M_1 and M_2 satisfy our requirements. First we show that $|M_1| + |M_2| \leq 3m$. By Claim 4.1, $|M_2| = k + m$ and $|M_1| = \text{size}((L_2\#^m)^+) = \text{size}(L_2) + m$. Thus $|M_1| + |M_2| = k + m + \text{size}(L_2) + m = 3m$. Next we show that $L(M_1) \cup L(M_2) = L_5$.

$$\begin{aligned} L(M_1) \cup L(M_2) &= (L_2\#^m)^+ \cup (L(M')\#^m) \\ &= (L_2\#^m)^+ \cup (L_2\#^m) \cup (L(M')\#^m) && (\text{since } L_2\#^m \subseteq (L_2\#^m)^+) \\ &= (L_2\#^m)^+ \cup (L_2 \cup L(M'))\#^m \\ &= (L_2\#^m)^+ \cup (L_1\#^m) && (\text{by Claim 4.2}) \\ &= L_3 \cup L_4 \\ &= L_5. \end{aligned}$$

(if) Suppose L_5 can be expressed as the union of $L(M_1)$ and $L(M_2)$ for some DFA's M_1 and M_2 such that $|M_1| + |M_2| \leq 3m$. We can assume without loss of generality that both $L(M_1)$ and $L(M_2)$ are nonempty. (If not, let $L(M_1) = \emptyset$. Then $L(M_2) = L_5$, and we can show that $|M_2|$ must be at least m^2.) We further assume that M_1 and M_2 are minimized. We then make the following claim.

CLAIM 4.4. *Either* (i) $L(M_1) \supseteq S\#^m$ *and* $L(M_2) = L_3$ *or* (ii) $L(M_2) \supseteq S\#^m$ *and* $L(M_1) = L_3$.

Proof. Informally, the idea behind the proof can be outlined as follows. Since $L(M_1) \cup L(M_2) = L_5$, the two DFA's M_1 and M_2 should collectively have a #-tail (a sequence of m #-arcs ending at some state with no outgoing arc) and a #-waist (a sequence of m #-arcs ending at some state with at least one outgoing arc). Since $L(M_1)$ and $L(M_2)$ are nonempty, both must contain either a #-tail or a #-waist. But if one of them contains both a #-tail and a #-waist, then $|M_1| + |M_2|$ will exceed $3m$. Thus exactly one of them contains a #-tail and the other contains a #-waist. We can then show that the DFA containing the #-tail must accept every string in $S\#^m$ and that the DFA containing the #-waist must accept the language L_3.

Formally, a #-tail is defined as a sequence of states $q_1, q_2, \ldots, q_m$ such that (i) $\delta(q_i, \#) = q_{i+1}$ for $i = 1, 2, \ldots, m-1$ and (ii) q_m has no outgoing arc. A #-waist is defined as a sequence of states $q_1, q_2, \ldots, q_m$ such that the following conditions hold: (i) for all $i = 1, 2, \ldots, m-1$, $\delta(q_i, \#) = q_{i+1}$ and (ii) q_m has at least one outgoing arc. Claim 4.4 is an easy consequence of the following observations, which we state without proof:

Observation 4.1. M_i must contain a #-tail or a #-waist (or both).

Observation 4.2. If either M_1 or M_2 contains both a #-tail and a #-waist, then $|M_1|+|M_2| > 3m$.

Observation 4.3. If M_i contains the #-tail, then $L(M_i) \supseteq S\#^m$. (To prove this observation, note that $L_2 \cap S = \emptyset$. Thus the DFA containing the #-waist does not accept any string in $S\#^m$.)

Observation 4.4. If M_i contains the #-waist, then $L(M_i) = L_3$.

This concludes the proof of Claim 4.4. □

Suppose M_2 contains the #-tail (the other case is identical). Then $S\#^m \subseteq L(M_2) \subseteq L_4$, i.e., $S \subseteq L(M_2)/\#^m \subseteq L_1 = \overline{T}$. Thus L_6 defined as $L_6 = L(M_2)/\#^m$ is a regular language consistent with S and T.

The rest of the proof of Theorem 4.3 is aimed at showing that a k-state DFA accepting L_6 can be designed from M_2 by removing the #-tail from it. We do this by showing that the size of M_2 is at most $m + k$.

CLAIM 4.5. $|M_2| \leq m + k$.

Proof. Observe that $|M_2| \leq 3m + 1 - |M_1|$. We also note that $|M_1| \geq \text{size}(L_3)$. (This is true because $L(M_1) = L_3$ and M_1 is minimized.) Hence

$$\begin{aligned} |M_1| &= \text{size}(L_3) \\ &= \text{size}((L_2\#^m)^+) \\ &= m + \text{size}(L_2) && \text{(by Claim 4.1)} \\ &= 2m - k && \text{(by definition of } m). \end{aligned}$$

Thus $|M_2| \leq 3m - (2m - k) = m + k$. □

Thus the DFA M accepting $L_6 = L(M_2)/\#^m$ has at most k states since all states in the #-tail of M_2 have been removed. This concludes the proof of Claim 4.3 and Theorem 4.3. □

We next consider concatenation and Kleene star. We show that the problems corresponding to both of these operations are PSPACE-complete. We need the following lemma 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.*

Proof. Let $L_3 = \{x1y | x \in (0+1)^*, |y| = 2\log_2 n\}$. The claim follows since $L_3 = L_4 \cdot L_5$, where $L_4 = (0 + 1)^*1$ and L_5 is the set of all strings of length $2\log_2 n$. We omit the easy proof that conditions (i) and (ii) are satisfied. □

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

Proof. MCG is easily seen to be in PSPACE. In what follows we will prove its PSPACE-hardness by reducing CE to MCG in polynomial time. (Recall that CE was shown to be PSPACE-complete in Theorem 4.1.) Let M_1, M_2, M_3 be three DFA's forming an instance of the CE problem. We may assume without loss of generality that $L(M_3) = \Sigma^*$ and that M_1 and M_2 are minimal. Let $n = |M_1| + |M_2|$, let $L_1 = L(M_1)$, and let $L_2 = L(M_2)$. Let 0 and 1 be two new symbols not in Σ. Let L_3, L_4, and L_5 be as in Lemma 4.1, and let # be another new symbol.

Now let

$$L = (L_1\#L_3\#L_2) \cup (L_1\#L_4 \cdot L_2) \cup (L_1 \cdot L_5\#L_2) \cup \Sigma^*.$$

Let M be the minimum DFA accepting L. It can be shown that $|M| = \Theta(n^2)$ and that M can be constructed from M_1 and M_2 in polynomial time. The reduction is complete with M and $k = 3n$ forming an instance of the MCG problem. We complete the proof by establishing the correctness of the reduction:

CLAIM 4.6. *$L_1 \cdot L_2 = \Sigma^*$ if and only if there exist* DFA's *N_1, N_2 such that $L(N_1) \cdot L(N_2) = L$ and $|N_1| + |N_2| \leq k$.*

Proof. (only if) Let $L(N_1) = L_1(\#L_4 \cup \{\epsilon\})$, let $L(N_2) = (L_5\# \cup \{\epsilon\})L_2$, and choose N_1 and N_2 to be minimal. It is easy to show that $|N_1| \leq n + 2$ and that $|N_2| \leq n + 2\log_2 n + 1$, so that $|N_1| + |N_2| \leq 2n + 2\log_2 n + 3 \leq 3n$ for large enough n.

(if) It is easy to see that if $L(N_1)$ contains any string in $\Sigma^*\#(0 + 1)^+\#$, then $L(N_2)$ can only have strings over Σ. Thus one of the following must be true: (i) $L(N_i) \subseteq \Sigma^*$ for either $i = 1$ or $i = 2$ or (ii) $L(N_1) \subseteq (\Sigma^*\#(0 + 1)^*) \cup \Sigma^*$ and $L(N_2) \subseteq ((0 + 1)^*\#\Sigma^*) \cup \Sigma^*$.

Suppose that (i) is true and that $L(N_1) \subseteq \Sigma^*$ (the proof for $L(N_2) = \Sigma^*$ is identical). We can show in this case that $|N_2| \geq \Omega(n^2)$: The idea is to note that we can get a DFA for L_3 from N_2 by removing some (but not adding any) states. We omit the details.

Thus (ii) must be true. Let

$$L_6 = L(N_1) \cap \Sigma^*,$$
$$L_7 = L(N_2) \cap \Sigma^*.$$

Then clearly $L_6 \cdot L_7 = \Sigma^*$. Also,

$$L_6 \cdot (L(N_2) \cap ((0+1)^*\#\Sigma^*)) = L_1 \cdot L_5\#L_2,$$

$$(L(N_1) \cap (\Sigma^*\#(0+1)^*)) \cdot L_7 = L_1\#L_4 \cdot L_2.$$

Since neither L_4 nor L_5 is empty, it follows that $L_6 = L_1$ and $L_7 = L_2$. Thus $L_1 \cdot L_2 = \Sigma^*$. This concludes the proofs of Claim 4.6 and Theorem 4.4 ☐

We next consider Kleene star. The Minimum Kleene Star Generation (MKSG) problem is also PSPACE-complete. The proof is similar to that of Theorem 4.2, but it is more complicated.

THEOREM 4.5. *The Minimum Kleene Star Generation problem is* PSPACE-*complete.*

Proof. The reduction is from SCE. Let the instance of SCE be M_1 and M_2 (over alphabet Σ) such that $L(M_1)$ and $L(M_2)$ do *not* contain ϵ. Recall that we want to decide whether $L(M_1) \cdot L(M_2) = L_0 = \{x | x \in \Sigma^*, |x| \geq 2\}$. Let $n = |M_1| + |M_2|$.

Let 0, 1, and # be new symbols not in Σ, and let $\Gamma = \{0, 1, \#\} \cup \Sigma$. Let L_3, L_4, and L_5 be as in Lemma 4.1. Note that L_5 is finite.

We now define the following new languages:

$$\begin{aligned}
L_6 &= L(M_2) \cdot L_4,\\
L_7 &= \#L(M_1),\\
L_8 &= L_5\#,\\
L_9 &= \Gamma^*(0+1)\Sigma\Gamma^*,\\
L_{10} &= \Gamma^+\#\Gamma^+,\\
L_{11} &= (\#L_0 \cdot L_4) \cup (L(M_2) \cdot L_3\#) \cup (\#L_0 \cdot L_3\#),\\
L_{12} &= L_6 \cup L_7 \cup L_8 \cup L_9 \cup L_{10},\\
L_{13} &= L_6^* \cup L_7 \cup L_8 \cup L_9 \cup L_{10} \cup L_{11}.
\end{aligned}$$

We state and prove the following claims about these languages:

CLAIM 4.7. *If* $L(M_1) \cdot L(M_2) = L_0$, *then* $L_{12}^* = L_{13}$.

CLAIM 4.8. *The minimum* DFA M *accepting* L_{13} *has at least* $\Omega(n^2)$ *states, and it can be designed in polynomial time.*

CLAIM 4.9. *There is a* DFA M *with* $25n$ *states that accepts* L_{12}.

Proof of Claim 4.7. Suppose $L(M_1) \cdot L(M_2) = L_0$. Then it can be observed that $L_{13} = L_7 \cdot L_6 \cup L_6 \cdot L_7 \cup L_7 \cdot L_6 \cdot L_8$, and so it follows that $L_{12}^* \subseteq L_{13}$. The reverse inclusion is more complicated. Let $w \in L_{12}^*$. We will show that $w \in L_{13}$. We have to consider many cases. (Assume that $w \neq \epsilon$.)

Case 1: w contains three or more occurrences of #. Then $w \in L_{10}$, and so $w \in L_{13}$.

Case 2: $w \in L_6^*$, i.e., w was obtained by repeated use of L_6 alone. Then obviously $w \in L_{13}$.

Case 3: w has a substring in L_9 or L_{10}. In this case it is easily seen that $w \in L_9$ or $w \in L_{10}$ and hence in L_{13}.

If none of these cases applies, then w should satisfy the following conditions: (a) w begins or ends with a # symbol, (b) w has at most two occurrences of #, and (c) $w \in (L_6 \cup L_7 \cup L_8)^* - L_6^*$. Now it is easy to see that w should be in $L_7L_6^* \cup L_6^*L_8 \cup L_7L_6^*L_8$. Since $L_7L_6^* \cup L_6^*L_8 \cup L_7L_6^*L_8 \subseteq L_{11} \subseteq L_{13}$, the claim follows. ☐

Proof of Claim 4.8. The lower bound on the size of M follows from the observation that a DFA for L_3 can be obtained from a DFA for L_{13} by removing (but not adding) states. We leave out the easy proof that M can be designed in polynomial time. □

We omit the proof of Claim 4.9.

The reduction is complete with the DFA M for L_{13} and $k = 25n$ forming the instance of the MKSG problem. Obviously the reduction can be carried out in polynomial time, as noted in Claim 4.8. We prove the correctness of the reduction:

CLAIM 4.10. $L(M_1) \cdot L(M_2) = L_0$ *if and only if there is a* DFA M *with k states such that* $L_{13} = L(M)^*$.

Proof. (only if) This readily follows from the observations made above. One choice of M is the DFA of Claim 4.9.

(if) This is more involved. We outline a sketch of the proof. Since neither $L(M_2)$ nor L_4 contains ϵ, it is easy to see that

$$\begin{gathered} L(M) \cap (\Sigma^* \cup (0+1)^*) = \emptyset, \\ L(M) \cap (\Sigma^+ \cdot (0+1)^+) = L_6, \\ L(M) \cap (\#\Sigma^*) = L_7, \\ L(M) \cap (\#\Sigma^* \cdot (0+1)^*) = L_7 \cdot L_6, \\ L(M) \cap ((0+1)^*\#) = L_8, \\ L(M) \cap (\Sigma^*(0+1)^*\#) = L_6 \cdot L_8. \end{gathered}$$

Let $L_{14} = L(M) \cap (\#\Sigma^*(0+1)^*\#)$, and let L_{15} be obtained by homeomorphically erasing the symbols of $\Sigma \cup \{\#\}$ from L_{14}. More formally, let $h : \Gamma \to \Gamma^*$ be defined as $h(a) = a$ for $a \notin \Sigma \cup \{\#\}$, $h(a) = \epsilon$ if $a \in \Sigma \cup \{\#\}$. Then let $L_{15} = h(L_{14})$.

Clearly, $L_{15} \subseteq L_3$. In fact, we can show that $L_{13} \subset L_3$. (Otherwise, it follows that there is a k-state DFA accepting L_3 since we can obtain a k-state DFA for L_3 by suitably removing some (and not adding any) states from M.) Moreover, one can show that $L_3 - L_{13}$ is infinite. (Otherwise, one can still construct a k-state DFA for L_3 by removing states from M.)

Let x be a string in $L_3 - L_{13}$ such that x is longer than all strings in L_5. Note that $x \in (0+1)^+$. Clearly,

$$(\#L_0 \cdot x\#) \subseteq L_9 \subseteq L_{11} = L(M)^*.$$

It is easy to see that (again note that $L_6 \subseteq \Sigma^+ \cdot (0+1)^+$)

$$\begin{aligned} (\#L_0 \cdot x\#) &= L(M)^* \cap (\#L_0 \cdot x\#) \\ &= ((L_7 \cdot L_6 \cdot L_8) \cap (\#L_0 \cdot x\#)) \cup (L_7 \cdot L_8 \cap (\#L_0 \cdot x\#)) \cup (L_{14} \cap (\#L_0 \cdot x\#)) \\ &= ((L_7 \cdot L_6 \cdot L_8) \cap (\#L_0 \cdot x\#)) \cup (L_7 \cdot L_8 \cap (\#L_0 \cdot x\#)) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{since } L_{14} \cap (\#L_0 \cdot x\#) = \emptyset) \\ &= ((\#L(M_1) \cdot L(M_2) \cdot L_4 \cdot L_5\#) \cap (\#L_0 \cdot x\#)) \cup ((\#L(M_1) \cdot L_5\#) \cap (\#L_0 \cdot x\#)). \end{aligned}$$

Since x is longer than all strings in L_5,

$$(\#L(M_1) \cdot L_5\#) \cap (\#L_0 \cdot x\#) = \emptyset.$$

Thus $(\#L(M_1) \cdot L(M_2) \cdot L_4 \cdot L_5\#) \cap (\#L_0 \cdot x\#) = (\#L_0 \cdot x\#)$. That is, $L(M_1) \cdot L(M_2) = L_0$. This completes the proofs of Claim 4.10 and Theorem 4.5. □

We conclude §4.2 with the Minimum Reversal Generation (MRG) problem. The problem is to find, given a DFA M and an integer k, whether there is a DFA M' with at most k states such that $L(M) = L(M')^{\text{rev}}$ or, equivalently, $L(M)^{\text{rev}} = L(M')$. Let M^{rev} denote the reverse of M obtained by using the standard algorithm (see the beginning 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 §3. We show that the 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 §3, that the more general problem of finding whether there is a k-stage DFA equivalent to a given NFA is NP-hard even if k is fixed.

The MRG problem can 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 (with respect to 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 that we can define the diversity of a regular language L as the diversity of DFA accepting L. It has been observed by Young and 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, whether the diversity of $L(M)$ is bounded by k.

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

Proof. Let M, a DFA, and integer k be given as inputs. Let $M_1 = M^{\text{rev}}$. We apply the standard algorithm for converting an NFA to a DFA by subset construction [Ho79] to M' with the condition that it produce only live subsets. We apply this algorithm on M' until the number of states generated exceeds k. In this case the algorithm outputs "no." If the conversion is complete with k or fewer subsets generated, the output is "yes." Clearly, the algorithm runs in time bounded by a polynomial in $k + |M|$. It is also clear that the "yes" answer of the algorithm is correct. We show that the "no" answers are also correct by showing that the resulting DFA produced by the subset construction method is, in fact, minimal when applied to a reverse deterministic NFA.

Let M be a DFA with states $Q = \{q_1, q_2, \ldots, q_n\}$. Define $L(q_i)$ as the set of strings accepted by M is q_i is considered as the start state and if there is no other change in M. In the same way define $P(q_i)$ as the set of strings accepted by M is q_i is considered the only accepting state with no other change in M. Let M_1 be the resulting DFA when the subset construction method is applied to M^{rev}. The states of M_1 are named by the subsets of Q. Let $T_1, T_2 \subseteq Q$ be two states of M_1. We observe that $L(T_i)$ can be expressed as

$$L(T_i) = \bigcup_{q \in T_i} P(q)^{\text{rev}}, \qquad i = 1, 2.$$

Since M is a DFA, $P(q_1), P(q_2), \ldots, P(q_n)$ are pairwise disjoint. It follows that $T_1 \neq T_2$ implies $L(T_1) \neq L(T_2)$. This concludes the proof. □

5. Conclusions.

The 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 of 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 §4 reveal that even the weakest forms of nondeterminism can render a decision problem intractable. These results substantially strengthen the known hardness results in which the input machines are generally assumed to have unrestricted nondeterminism. Although our proofs use the classical technique of reduction, the reductions and the connections we have established are new; they may find wider applications in the study of other related problems.

In a companion paper [Ji90] we study the DFA $\rightarrow$ NFA problem over a unary alphabet. It turns out that this problem is hard even in this special case. However, the evidence of hardness we can provide is weaker than that for the standard ones, such as NP-hardness and PSPACE-hardness. The result can be stated more precisely as follows: A *unary cyclic* DFA is a DFA over a unary alphabet, in which the states form a simple cycle. A unary cyclic language is a regular language accepted by a unary cyclic DFA. The main result of [Ji90] is, "[The] DFA $\rightarrow$ NFA problem, when the input is a unary cyclic DFA, is in NP but not in P unless every language in NP can be accepted by a deterministic Turing machine operating in $n^{O(\log n)}$ time." Whether this problem is NP-complete remains an interesting open question.

Our work has settled most of the fundamental questions in the proposed area of study. Yet there are some questions that are not resolved. Some of them are stated below:

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

(ii) Is the 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? (Clearly, there is an exponential increase in the size when a UFA is converted to a DFA, even when the input is restricted to a unary alphabet [Ra89].)

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

Appendix. We present below a list of significant decision problems relevant to this paper. This includes our new results as well as the known problems that we use in our reductions as candidates. We present them in the format used in the appendix of the book by Garey and Johnson [Ga78], which has become quite standard at present for stating decision problems. For each problem we also cite the section and the result where it appears. When presenting the results from other published work, we also cite the original reference, along with its use in our work.

Normal set basis

INSTANCE: Collection C of subsets of a finite set S and positive integer $k \leq |C|$.

QUESTION: Is there a collection B of subsets of S with $|B| \leq k$ such that for each $c \in C$ there is a pairwise disjoint subcollection of B whose union is exactly c?

Remark. The problem is NP-complete, as shown in Lemma 3.3. Stockmeyer [St76] earlier showed that the Set Basis problem is NP-complete.

Universe problem for multiple DFA

INSTANCE: A collection of DFA's $M_1, \ldots, M_n$ over a finite alphabet Σ.

QUESTION: Is $\bigcup_i L(M_i) = \Sigma^*$?

Remark. The problem is PSPACE-complete. The proof readily follows from the next result. This result is used in the proof of Theorem 3.2

Finite automata intersection (Theorem 3.2)

INSTANCE: A collection of DFA's $M_1, \ldots, M_n$ over a finite alphabet Σ.

QUESTION: Is $\bigcap_i L(M_i) = \emptyset$?

Remark. The problem is PSPACE-complete [Ko77]. This result is used in the above problem.

Vertex cover

INSTANCE: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

QUESTION: Is there a *vertex cover* of size k or less for G, that is, a subset $V' \subseteq V$ such that $|V| \leq k$ and, for each edge $\langle u, v\rangle \in E$ either u or v (or both) belongs to V'?

Remark. The problem is NP-complete [Ka72]. This result is used in the proof of Lemma 3.3.

DFA → NFA (UFA)

INSTANCE: A DFA M and an integer k.

QUESTION: Is there an NFA (UFA) with k (or fewer) states accepting $L(M)$?

Remark. The problem is PSPACE-complete, as shown in Theorem 4.2. The problem remains PSPACE-complete when the alphabet is binary. The unary case is partially solved in [Ji90]. In Theorem 4.2 the problem DFA → UFA is shown to be NP-complete.

Linear space acceptance

INSTANCE: A linear space-bounded deterministic Turing machine M and a string w.

QUESTION: Is $w \in L(M)$?

Remark. The problem is PSPACE-complete [Ka72]. This result is used to prove Theorem 4.1.

Concatenation equivalence (CE)

INSTANCE: DFA's M_1, M_2, and M_3.

QUESTION: Is $L(M_1) \cdot L(M_2) = L(M_3)$?

Remark. The problem is PSPACE-complete, as shown in Theorem 4.1. This result has been used to derive a PSPACE-completeness result in Theorem 4.4.

Strong concatenation equivalence (SCE)

INSTANCE: DFA's M_1, M_2.

QUESTION: (Is $L(M_1) \cdot L(M_2) = \{x | x \in \Sigma^*, |x| \geq 2\}$?) Is $L(M_1) \cdot L(M_2) = L(M_3)$?

Remark. The problem is PSPACE-complete, which follows by a minor modification of the proof of Theorem 4.1. This result is used to prove a PSPACE-hardness result in Theorem 4.5.

Kleene star equivalence (KSE)

INSTANCE: DFA's M_1 and M_2.

QUESTION: Is $L(M_1)^* = L(M_2)$?

Remark. The problem is PSPACE-complete, as shown in Theorem 4.2. The reduction is from SCE.

Minimum inferred DFA [Go78]

INSTANCE: Finite alphabet Σ, two finite subsets $S, T \subseteq \Sigma^*$, and a positive integer k.

QUESTION: Is there a k-state DFA that accepts a language L such that $S \subseteq L$ and $T \subseteq \Sigma^* - L$?

Remark. The problem is NP-complete [Go78]. This result is used to prove Theorem 4.3.

Minimum Op generation (Op = union, concatenation, Kleene star, or reversal)

INSTANCE: DFA M and an integer k.

QUESTION: Are there DFA's M_1 and M_2 (DFA M_1 if Op is unary) with $|M_1| + |M_2| \leq k$ ($|M_1| \leq k$) such that $L(M_1)\text{Op}L(M_2) = L(M)$? ($\text{Op}(L(M_1)) = L(M)$?)

Remark. The problems are NP-complete for union and intersection, PSPACE-complete for concatenation and Kleene star, and solvable in pseudo-polynomial time for reverse.

Acknowledgments. We are very grateful to Wing Ning Li for once again bringing the DFA → NFA problem to our attention. We 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]. We thank the three anonymous referees, whose valuable suggestions improved the presentation of this paper.

REFERENCES

[Ch86] M. CHROBAK, *Finite automata and unary languages*, Theoret. Comput. Sci., 47 (1986), pp. 149–158.

[Co71] S. COOK, *Complexity of theorem proving procedures*, in Proc. 3rd Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1971, pp. 151–158.

[Ga78] M. GAREY AND D. JOHNSON, *Computers and Intractability: A Guide to* NP-*Completeness*, Freeman, San Francisco, 1978.

[Gi68] A. GINZBURG, *Algebraic Theory of Automata*, Academic Press, New York, 1968.

[Go78] E. GOLD, *Complexity of automaton identification from given data*, Inform. and Control, 37 (1978), pp. 302–320.

[Ha66] J. HARTMANIS AND R. STEARNS, *Algebraic Structure Theory of Sequential Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[Ho71] J. HOPCROFT, *An $n \log n$ algorithm for minimizing the states in a finite automation*, in The Theory of Machines and Computations, Z. Kohavi, ed., Academic Press, New York, 1971, pp. 189–196.

[Ho79] J. HOPCROFT AND J. ULLMAN, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979.

[Hu73] H. HUNT, *On the time and tape complexity of languages*, in Proc. 5th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1973, pp. 10–19.

[Hu74] H. HUNT AND D. ROSENKRANTZ, *Computational parallels between regular and context-free languages*, in Proc. 6th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1974, pp. 64–74.

[Hu76] H. HUNT, D. ROSENKRANTZ, AND T. SZYMANSKI, *On the equivalence, containment and covering problems for the regular and context-free languages*, J. Comput. System Sci., 12 (1976), pp. 222–268.

[Ji90] T. JIANG, E. MCDOWELL, AND B. RAVIKUMAR, *The structure and complexity of minimal* NFA*'s over unary alphabet*, Internat. J. Found. Comput. Sci., 2 (1991), pp. 163–182.

[Ka72] R. KARP, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum, New York, 1972, pp. 85–103.

[Ko77] D. KOZEN, *Lower bounds for natural proof systems*, in Proc. 18th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1977, pp. 254–266.

[Ma78] A. MANDEL AND I. SIMON, *On finite semi-groups of matrices*, Theoret. Comput. Sci., 5 (1978), pp. 183–204.

[Me71] A. MEYER AND M. FISCHER, *Economy of description by automata, grammars and formal systems*, in Proc. 12th Annual IEEE Symposium on Switching and Automata Theory, IEEE Computer Society, Washington, DC, 1971, pp. 188–191.

[Me72] A. MEYER AND L. STOCKMEYER, *The equivalence problem for regular expressions with squaring requires exponential space*, in Proc. 13th Annual IEEE Symposium on Switching and Automata Theory, IEEE Computer Society, Washington, DC, 1972, pp. 125–129.

[My57] J. MYHILL, *Finite automata and the representation of events*, Tech. Report WADD TR-57-624, Wright Patterson AFB, OH, 1957, pp. 112–137.

[Na91] B. NATARAJAN, *Machine Learning, A Theoretical Approach*, Morgan-Kaufmann, San Mateo, CA, 1991.

[Ne58] A. NERODE, *Linear automaton transformations*, Proc. Amer. Math. Soc., 9 (1958), pp. 541–544.

[Pi89] L. PITT AND M. WARMUTH, *The minimum consistent* DFA *problem cannot be approximated within any polynomial*, in Proc. 21st Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1989, pp. 421–432.

[Ra59] M. RABIN AND D. SCOTT, *Finite automata and their decision problems*, IBM J. Res. Develop., 3 (1959), pp. 114–125.

[Ra89] B. RAVIKUMAR AND O. IBARRA, *Relating the type of ambiguity to the succinctness of their representations*, SIAM J. Comput., 18 (1989), pp. 1263–1282.

[Re77] C. REUTENAUER, *Propriétés arithmétiques et topologiques de séries rationelles en variables noncommutatives*, Thèse troisième cycle, Universite de Paris VI, Paris, 1977.

[Ri87] R. RIVEST AND R. SCHAPIRE, *Diversity-based inference of finite automata*, in Proc. 28th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1987, pp. 78–87.

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

[Sc88] R. SCHAPIRE, *Diversity Based Inference on Finite Automata*, master's thesis, MIT Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1988; Tech. Report MIT/LCS/TR-413, 1988.

[St85] R. STEARNS AND H. HUNT, *On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata*, SIAM J. Comput., 14 (1985), pp. 598–611.

[St73] L. STOCKMEYER AND A. MEYER, *Word problems requiring exponential time (preliminary report)*, in Proc. 5th Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, New York, 1973, pp. 1–9.

[St76] L. STOCKMEYER, *Set Basis Problem is* NP-*Complete*, Report RC-5431, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1976.

[Th68] K. THOMPSON, *Regular expression searching algorithm*, Comm. ACM, 11 (1968), pp. 419–422.

[Tz89] W. TZENG, *The equivalence and learning of probabilistic automata*, in Proc. 30th Annual IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society, Washington, DC, 1989, pp. 268–273.