The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial

LEONARD PITT

University of Illinois, Urbana, Illinois

AND

MANFRED K. WARMUTH

University of California, Santa Cruz, California

Abstract. The minimum consistent DFA problem is that of finding a DFA with as few states as possible that is consistent with a given sample (a finite collection of words, each labeled as to whether the DFA found should accept or reject). Assuming that P $\neq$ NP, it is shown that for any constant k, no polynomial-time algorithm can be guaranteed to find a consistent DFA with fewer than opt^k states, where opt is the number of states in the minimum state DFA consistent with the sample. This result holds even if the alphabet is of constant size two, and if the algorithm is allowed to produce an NFA, a regular expression, or a regular grammar that is consistent with the sample. A similar nonapproximability result is presented for the problem of finding small consistent linear grammars. For the case of finding minimum consistent DFAs when the alphabet is not of constant size but instead is allowed to vary with the problem specification, the slightly stronger lower bound on approximability of $opt^{(1-\epsilon)\log\log opt}$ is shown for any $\epsilon > 0$.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*automata*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*reducibility and completeness*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems—*decision problems*

General Terms: Algorithms, Languages, Theory

Additional Key Words and Phrases: Approximation algorithms; minimization of finite state machines; nonapproximability

---

The work of L. Pitt was supported in part by National Science Foundation (NSF) grant IRI 88-09570, and by the Department of Computer Science, University of Illinois at Urbana-Champaign.

The work of M. K. Warmuth was supported by Office of Naval Research grants N00014-86-K-0454 and N0014-85-K-0445. Part of this work was done while M. K. Warmuth was on leave from University of California, Santa Cruz, at Aiken Computation Laboratory, Harvard University.

Author's addresses: L. Pitt, Department of Computer Science, University of Illinois, Urbana, IL 61801; M. K. Warmuth, Department of Computer and Information Sciences, University of California at Santa Cruz, Santa Cruz, CA 95064.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1993 ACM 0004-5411/93/0100-0095 $01.50

1. *Introduction*

We consider the following problem. Given finite sets POS and NEG of words over a finite alphabet, can a small deterministic finite automaton (DFA) be constructed that is consistent with POS and NEG, that is, accepts all words of POS and rejects all words of NEG? It is known that the problem of determining the smallest such consistent DFA for a given sample is NP-hard [11], and thus is unlikely to be solvable with a polynomial-time algorithm [10]. It is natural to ask whether an *approximately small* DFA can be found: Is there an efficient algorithm, that for some reasonably slowly growing function f, can produce (or just determine the existence of) a consistent DFA of size $f(opt)$, where opt is the size of the smallest consistent DFA?

We answer this question negatively by proving that, assuming $\mathrm{P} \neq \mathrm{NP}$, there does not exist a polynomial-time algorithm A and constant k such that on input of any finite sets of strings POS and NEG (over alphabet $\{0,1\}^*$), A outputs a *nondeterministic* finite automaton (NFA) that is consistent with POS and NEG, and has less than opt^k states, where opt is the minimum number of states of any consistent DFA.

It follows that unless $\mathrm{P} = \mathrm{NP}$, no element of any of the naturally used representations of the regular sets (DFAs, NFAs, regular expressions, or regular grammars) can always be found that is of size at most polynomially larger than the smallest DFA consistent with a sample over a two-letter alphabet. This significantly improves the lower bound on approximability due to Li and Vazirani [18], which shows that a constant factor of $\frac{9}{8}$ cannot be achieved.

The same techniques are used to also show that the linear grammar consistency problem cannot be approximated within any polynomial factor unless $\mathrm{P} = \mathrm{NP}$. More specifically, given two finite sets POS and NEG consistent with some linear grammar G, it is NP-hard to find a linear grammar G' that generates all of the strings of POS, none of the strings of NEG, and has size bounded by some polynomial in the size of G.

An interesting extension of our results is that when the alphabet is allowed to vary (i.e., when the alphabet is considered as part of the problem specification), then unless $\mathrm{P} = \mathrm{NP}$, no polynomial-time approximation algorithm can determine if there exists a consistent DFA or NFA with at most $opt^{(1-\epsilon)\log\log opt}$ states, or a consistent regular grammar or regular expression with at most $opt^{(1-\epsilon)\log\log opt}$ symbols, where opt is the number of states of the minimum state consistent DFA, and ϵ is any positive constant.

1.1. Other Nonapproximability Results.

There seem to be few naturally arising optimization problems for which nonapproximability results have been shown. Indeed, the dearth of such results is one of the motivations given in a number of recent papers for the investigation of approximation preserving reductions [17, 20, 21]. The traveling salesperson problem (TSP) is perhaps the most notable optimization problem that cannot be approximated (in the absence of other constraints, e.g., triangle inequality) [10] assuming $\mathrm{P} \neq \mathrm{NP}$. However, the reason that TSP is not approximable is that it is essentially the weighted version of the NP-complete Hamiltonian cycle problem. Although one may similarly define *optimization* problems based on other NP-complete *decision* problems in such a way that the optimization problem cannot be approximated at all (or at least not very well), such results are typically uninteresting for two reasons—the problems defined usually are not natural, and the resulting proofs are trivial. In contrast, the minimum consistent DFA problem discussed here is a natural problem, and the nonapproximability result is not obtained by simply adding weights to an NP-complete decision problem.

Besides TSP, among the seemingly few existing negative approximability results, two others are well known, but the bounds are much weaker than those shown for TSP and the result given here for DFAs. For minimum graph coloring [9], it was shown that (unless P = NP) no polynomial-time approximation algorithm exists guaranteeing a constant factor approximation strictly smaller than twice optimal. Also, for maximum independent set (equivalently, maximum clique), it has been shown that if *some* constant factor approximation can be achieved, then *any* constant factor approximation can be achieved [10].

1.2. The Minimum Consistent DFA Problem. Gold [11] proved that the problem of finding a smallest consistent DFA is NP-hard. D. Angluin (private communication) showed that it is NP-hard to determine whether there exists a two-state DFA consistent with given data. Trakhtenbrot and Barzdin [24] gave a polynomial-time algorithm for finding a smallest consistent DFA in the case where the sets POS and NEG together consist of all strings up to a given length. Angluin [5] extended Gold's result, and showed that if even some small fraction ϵ of strings up to a given length were missing from POS $\cup$ NEG, then the problem is again NP-hard, and also showed that the problem of finding the smallest regular expression consistent with a finite sample is NP-hard. Angluin [2] left as an open question whether an approximately small DFA could be found.

In 1987, Li and Vazirani [18] gave the first nonapproximability result for the minimum consistent DFA problem, showing that if $P \neq NP$, no polynomial-time algorithm can find a consistent NFA of size smaller than $\frac{9}{8}$ times the size of a smallest consistent DFA. Our main theorem (Theorem 6.1) strengthens this result by replacing the constant factor $\frac{9}{8}$ with any polynomial function of optimal.

Finally, concurrent with this research, Kearns and Valiant give even stronger nonapproximability results for the minimum consistent DFA problem than the one presented here [16]. However, their results rely on cryptographic assumptions (e.g., that factoring Blum integers is intractable), whereas our results assume only that $P \neq NP$. We present a discussion of their work, and its relationship to ours, in Section 9.

It is important to note the distinction between the minimum consistent DFA problem, and the DFA state minimization problem. In the latter problem, the input is a DFA and the goal is to produce a DFA accepting the same language with a minimum number of states; this problem has well-known polynomial-time algorithms [15]. An obvious first attempt at solving the minimum consistent DFA problem is to create a DFA that accepts exactly the (finite) language POS (and no other strings), and then use the DFA state minimization algorithm to obtain a minimum state DFA for the language POS. However, it is possible that a much smaller DFA exists that accepts a superset of POS and no string of NEG. The minimum consistent DFA problem addresses the complexity of finding a regular language that separates POS from NEG and for which there is a small DFA.

1.3. IMPLICATIONS FOR LEARNING DFAs. Although our results mainly have significance in the context of combinatorial optimization, our original motivation was in the study of the learnability of DFAs from randomly generated examples in the distribution independent model of learning (now called "pac"-learning) introduced by Valiant [25]. By results from [6], if in fact there *was* a polynomial-time algorithm that could, given two finite sets POS and NEG, produce a consistent DFA of size at most polynomially larger than the smallest consistent DFA, then DFAs would be pac-learnable. This is only a sufficient condition for learnability; consequently our results *do not* show that DFAs are not pac-learnable.[1] However, our results *do* show that any efficient algorithm for learning DFAs would have to produce very large hypotheses (unless P = NP). Further discussion of the learnability of DFAs, the problem of finding a small consistent DFA, and the relationship between our work and recent work of Kearns and Valiant [16] (showing nonlearnability of DFAs based on cryptographic assumptions) is given in Section 9.

The rest of this paper is organized as follows: In Section 2, we introduce some basic definitions and techniques to be used throughout the paper. In Section 3, it is proved that no polynomial-time algorithm can guarantee a quadratic approximation for the minimum consistent DFA problem (unless P = NP). One of the reasons for including the quadratic case in the paper is that it gives some intuition for the polynomial nonapproximability results that follow in Section 4. We show that (unless P = NP) there is no polynomial-time algorithm for finding a consistent NFA of size polynomially larger than the size of the smallest consistent DFA (Theorem 4.1). In Section 5, we strengthen these results and show that an approximation of $opt^{(1-\epsilon)\log\log opt}$ cannot be guaranteed in polynomial-time for any positive constant ϵ unless P = NP. In these theorems the alphabet size of the DFA may vary with the problem instance. In Section 6 we turn our attention to the problem of finding small consistent DFAs over the two letter alphabet $\{0, 1\}$, and again prove that no polynomial-time approximation within a polynomial function of optimal is possible even in this simpler context (Theorem 6.1). Section 7 extends Theorem 6.1 to the cases when the approximation algorithm is allowed to output a regular expression or a regular grammar. Similar techniques are applied in Section 8 to show that the linear grammar consistency problem has no polynomial-time approximation algorithm. In Section 9, we discuss the relationship between approximability and learnability, noting different ways of measuring approximation performance, and relating this work to the recent results of [16]. It turns out that, with respect to certain performance measures, the proof for the quadratic case leads to a stronger nonapproximability result than the proof for the polynomial case. Finally, we conclude in Section 10 with some general comments and open problems.

2. *Definitions*

2.1. REPRESENTATIONS OF REGULAR LANGUAGES. In this section, we recall the standard definitions and basic facts about regular and linear languages. The reader unfamiliar with this material should consult [15] for further

[1]Angluin [4] has shown that DFAs are not learnable if the learner may only ask *equivalence queries* instead of receiving randomly generated examples. This result has no bearing on the optimization problem considered in this paper.

terminology and definitions. Σ is a finite alphabet, and Σ^* denotes all *words* (or *strings*) of finite length formed from the symbols of Σ. If $w \in \Sigma^*$, then $|w|$ denotes the number of symbols of w, and is called the *length* of w. The empty word λ is the unique word with length 0 in Σ^*.

Definition 2.1. A *deterministic finite automaton* (DFA) A is a 5-tuple $(Q, \Sigma, \delta, s_{\text{init}}, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $s_{\text{init}} \in Q$ is the *initial state*, δ is the *transition function* that maps $Q \times \Sigma$ *to* Q, and $F \subseteq Q$ is the set of *accepting* or *final* states. A *nondeterministic finite automaton* is a 5-tuple with the same parameters except that the transition function maps $Q \times (\Sigma \cup \{\lambda\})$ to 2^Q, the power set of Q.

The *size* of A, denoted by $|A|$, is the number of states of A. We use the standard graph representation of an NFA in which the vertices are the states of the NFA, and in which there is a directed edge (or *transition*) labeled with $\sigma \in \Sigma \cup \{\lambda\}$ from state s to state t if $t \in \delta(s, \sigma)$. Note that some edges may be labeled with λ. DFAs may be viewed as NFAs with the additional restriction that there are no λ-transitions, and for each state $s \in Q$ and letter $a \in \Sigma$, there is exactly one edge leaving s which is labeled with a.

A string $w \in \Sigma^*$ is *accepted* by the NFA (DFA) A iff there is a directed path leading from the initial state to some accepting state such that the concatenation of the symbols of the edges of the path forms the string w. (We say that the path is "labeled with" w.)

It is easy to show that for every NFA A, an NFA A' can be found in polynomial-time such that A' accepts the same language as A, A' has the same number of states as A, and A' has no λ-transitions [15]. Without loss of generality, we assume for the remainder of the paper that all NFAs have no λ transitions.

For any states s, t in Q, and any string w, we say w *leads* from s to t if there is a path labeled with w from s to t (In the case of a DFA, such a path is always unique). We also write w *leads to* t iff w leads from s_{init} to t.

A *positive example* of A is a word accepted by A and a *negative example* is a word in Σ^* that is not accepted by A. The *language accepted by* A, denoted by $L(A)$, is the set of all words accepted by A. The class of languages accepted by DFAs is identical to the class of languages accepted by NFAs and is called the class of *regular languages*. The name "regular" is derived from a third standard definition of regular languages in terms of *regular expressions*, defined below. For any regular expression r, $|r|$ denotes the *size* of the expression.

Definition 2.2. Let Σ be a finite alphabet. The *regular expressions over* Σ, the *size measure of regular expressions*, and the *languages that regular expressions denote* are defined recursively as follows:

(1) $\emptyset$ is a regular expression of size 1 denoting the empty language.
(2) For each $a \in \Sigma \cup \{\lambda\}$, the string a is a regular expression of size 1 denoting the language $\{a\}$.
(3) If r and s are regular expressions denoting the languages R and S, respectively, then $(r + s)$ and (rs) are regular expressions of size $|r| + |s| + 1$ denoting the languages $R \cup S$ and RS, respectively, and (r^*) is a regular expression of size $|r| + 1$ denoting the language R^*.

LEMMA 2.3. *For any regular expression r, there is an NFA A with at most $2|r|$ states such that $L(A)$ is the language denoted by r.*

Lemma 2.3 is proven constructively by induction on the depth of recursion in the definition of the expression r. The straightforward construction [15] makes use of λ-transitions, but as discussed above, these may be easily eliminated.

Another standard way to define languages is in terms of grammars. This leads to a fourth mechanism for defining regular languages.

Definition 2.4. A *context free grammar* G is a 4-tuple (Δ, Σ, P, S). Δ and Σ are disjoint finite sets of *nonterminals* and *terminals*, respectively. P is a finite set of *productions*; a production is of the form $A \to \alpha$ and has *size* $|A\alpha|$, where A is a nonterminal and $\alpha \in (\Delta \cup \Sigma)^*$. Finally, S is a special nonterminal called the *start symbol.*

A context free grammar is a *linear grammar* if all productions are of the form $A \to uBv$ or $A \to w$, where $u, v, w \in \Sigma^*$ and $B \in \Delta$. A linear grammar is *right linear* (*left linear*) if v is always λ (respectively, u is always λ). A *regular grammar* is either a right linear or a left linear grammar.

We associate a language with a context free grammar $G = (\Delta, \Sigma, P, S)$. If the production $A \to \beta$ is in P and α and γ are in $(\Delta \cup \Sigma)^*$, then $\alpha A \gamma$ *derives* $\alpha\beta\gamma$, written $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. Note that "$\Rightarrow$" defines a relation on words of $(\Delta \cup \Sigma)^*$. Let $\overset{*}{\Rightarrow}$ denote the reflexive and transitive closure of $\Rightarrow$. The *language generated by* G (denoted by $L(G)$) is given by $L(G) = \{w : w \in \Sigma^*$ and $S \overset{*}{\Rightarrow} w\}$.

It is easy to see that regular grammars generate exactly the class of regular languages. Let the size $|G|$ of a grammar G be the sum of the sizes of all of the productions.

LEMMA 2.5. *For any regular grammar G, there is an NFA A such that $|A| \leq 2|G|$ and $L(A) = L(G)$.*

PROOF. The proof is implicit in [15], and is sketched here for completeness. We first show how to construct an equivalent NFA A of size at most $2|G|$ from a right linear grammar G. Let the state/vertex set of A be the set of nonterminals that appear in some production of G. For any production $N_1 \to uN_2$, insert a directed edge labeled with u from N_1 to N_2. Add a special final state $\hat{F}$ to the state/vertex set and for any production $N \to w$ add a directed edge from state N to state $\hat{F}$ that is labeled with w. Convert the constructed graph into an NFA by replacing any edge that is labeled with a word of length r larger than 1 by a chain of r edges, each labeled with one letter. The chains are not allowed to have vertices in common. It is easy to see that the constructed NFA has at most $2|G|$ states, exactly one of which is final.

If G is left linear, then there is a corresponding right linear grammar of the same size that accepts $L(G)^R$, the language obtained by reversing all words of $L(G)$. Let A be an NFA of size at most $2|G|$ and with one final state that accepts $L(G)^R$. By swapping the initial and the final state, and reversing all of the transitions, an NFA of the same size that accepts $L(G)$ is obtained. □

Define a linear grammar to be *thin* if the number of symbols on the right side of each production is at most two. The following proposition is easily proved:

PROPOSITION 2.6. *For any linear grammar G, there is a thin linear grammar G' such that $|G'| \leq 3|G|$ and $L(G') = L(G)$.*

2.2. THE CONSISTENCY PROBLEM. We give some of the basic ideas and definitions that are used throughout the paper. A set of representations (encodings) of a class of languages $\mathscr{L}$ is a set $\mathscr{A}$ such that each $A \in \mathscr{A}$ denotes a language $L(A) \in \mathscr{L}$, and for each language $L \in \mathscr{L}$ there is at least one element of $\mathscr{A}$ that denotes L. Let $L(\mathscr{A}) = \{L(A)\colon A \in \mathscr{A}\}$ (thus, if $\mathscr{A}$ is a set of representations for $\mathscr{L}$, then $L(\mathscr{A}) = \mathscr{L}$). For example, the set of deterministic finite automata (DFAs) is a set of representations for the regular languages, as are NFAs, regular grammars, and regular expressions. We associate a size measure with each set of representations. The size (a nonnegative integer) of any element $A \in \mathscr{A}$ is denoted by $|A|$. The size measures for each of the sets of representations discussed in this section have already been defined.

Definition 2.7. A representation A is *consistent* with two sets of finite strings POS and NEG if POS is contained in $L(A)$ and NEG is disjoint from $L(A)$.

Definition 2.8. Let $\mathscr{A}$ and $\mathscr{B}$ be sets of representations of languages and let $L(\mathscr{A}) \subseteq L(\mathscr{B})$. The minimization problem MIN-CON($\mathscr{A}, \mathscr{B}$) is defined as follows:

Input instance: An instance I of MIN-CON($\mathscr{A}, \mathscr{B}$) consists of two finite sets of strings, POS and NEG, consistent with some element $A \in \mathscr{A}$.

Feasible solution: Any element $B \in \mathscr{B}$ that is consistent with I is a feasible solution. Note that there always exists a feasible solution.

Cost: The cost of a feasible solution B is the size $|B|$ of the representation B.

Optimal solution: For any instance I, the value $opt(I)$ is defined as the size of the smallest element of $\mathscr{A}$ that is consistent with I.

Note that a feasible solution of the problem requires a representation from the class $\mathscr{B}$, and optimality is defined with respect to elements of the class $\mathscr{A}$. Thus for some choices of $\mathscr{A}$ and $\mathscr{B}$, there may be no feasible solutions with cost as small as an optimal solution. (For example, consider MIN-CON(NFA, DFA)). Although the general definition assumes nothing regarding the relationship between the size of the smallest consistent members of $\mathscr{A}$ and $\mathscr{B}$, in this paper the latter is usually equal to or smaller than the former.

MIN-CON(DFA, NFA) is the main optimization problem we consider. This problem is easier than MIN-CON(DFA, DFA), since (1) every DFA is an NFA, and (2) in some cases, the smallest consistent NFA for a language is significantly smaller than the smallest consistent DFA. We prove hardness results for approximating MIN-CON(DFA, NFA), and thus for MIN-CON(DFA, DFA).

Definition 2.9. Let $\mathscr{A}, \mathscr{B}$ be sets of language representations, and f be any function of the single variable opt. Then *MIN-CON*($\mathscr{A}, \mathscr{B}$) is $f(opt)$-approximable iff there exists a constant c and a polynomial-time algorithm APPROX such that on input of any instance I of MIN-CON($\mathscr{A}, \mathscr{B}$) for which $opt(I) \geq c$, APPROX outputs a representation $B \in \mathscr{B}$ that is consistent with I and such that $|B| < f(opt(I))$.

Note that the definition of $f(opt)$-approximable does not require the approximation algorithm to perform well on all instances, but only on those instances I with sufficiently large values of $opt(I)$. Consequently, a result showing nonapproximability must show, for all approximation algorithms, that for all sufficiently large values of *opt* there are instances I with $opt(I) = opt$, and for which the approximation algorithm fails to achieve the desired bound. Thus, this is a stronger negative result than simply showing that the bound $f(opt)$ is not obtainable for particular values of *opt*.

The definition of MIN-CON($\mathscr{A}$, $\mathscr{B}$) depends on the (implicit) size measures used for $\mathscr{A}$ and $\mathscr{B}$. Our nonapproximability results are with respect to the particular size measures given in the previous subsection. However, the most important results of this paper are in the following form: MIN-CON($\mathscr{A}$, $\mathscr{B}$) is not $f(opt)$-approximable for any function f that is polynomially bounded. Such results are robust with respect to any size measure that is polynomially related to ours. However, in Section 5, we prove a slightly stronger nonapproximability result for MIN-CON(DFA, NFA) that holds when the size of a DFA or NFA is the number of states of the automaton, but does not hold when the size is the number of bits required to encode the automaton.

2.3. Using Gaps to Force Nonapproximability.

Our goal will be to show that, assuming $P \neq NP$, MIN-CON(DFA, NFA) is not $f(opt)$-approximable for any function f bounded above by some polynomial. Nonapproximability results may be obtained by exhibiting "gaps" in the cost measure for a minimization problem. Intuitively, if we can transform an instance of an NP-hard decision problem into a MIN-CON problem, such that if the answer to the NP-hard decision problem is "yes" then the optimal solution to the MIN-CON problem is some number p, whereas if the answer is "no," then there is no solution to the MIN-CON problem of size smaller than $f(p)$, then we can show that $f(opt)$-approximability of the MIN-CON problem implies that $P = NP$. More formally, we have the following sufficient condition:

Lemma 2.10. *Suppose there are infinitely many positive integers p such that there exists a polynomial-time transformation R_p with the following properties*:

Property 1. *R_p takes as input some instance I of an NP-complete language S, and outputs an instance of MIN-CON($\mathscr{A}$, $\mathscr{B}$).*

Property 2. *If instance $I \in S$, then $R_p(I)$ has an optimal solution with cost $opt(R_p(I)) = p$.*

Property 3. *If $I \notin S$, then $opt(R_p(I)) \geq f(p)$.*

Then, under the assumption that $P \neq NP$, MIN-CON($\mathscr{A}$, $\mathscr{B}$) is not $f(opt)$-approximable.

Proof. Suppose that the hypothesis of the lemma is true, and further suppose to the contrary that MIN-CON($\mathscr{A}$, $\mathscr{B}$) was $f(opt)$-approximable, witnessed by constant c and polynomial-time algorithm APPROX. We show that membership in S is decidable in polynomial time, hence $P = NP$, proving the lemma.

By hypothesis, there are infinitely many positive integers p such that the instance $R_p(I)$ of MIN-CON($\mathscr{A}$, $\mathscr{B}$) satisfies Properties 1–3 above. Choose any such value for p such that, in addition, $p \geq c$. Membership in S may be determined by the polynomial-time algorithm DECIDE: On input of any instance string I for which membership in S is to be determined, DECIDE computes $R_p(I)$ and gives this as input to subroutine APPROX. If APPROX returns a value of less than $f(p)$, then DECIDE outputs "$I \in S$." Otherwise, DECIDE outputs "$I \notin S$." Clearly, DECIDE runs in time polynomial in $|I|$, since both R_p and APPROX run in polynomial time.

To see that DECIDE determines membership in S correctly, observe that if DECIDE outputs "$I \in S$," then APPROX returned a value less than $f(p)$, which, by Property 3, implies that $I \in S$. Conversely, suppose that $I \in S$. Then, by Property 2, $R_p(I)$ has an optimal solution with cost p. By choice of p, $opt(R_p(I)) = p \geq c$. Since APPROX achieves the bound of f for all instances with $opt(R_p(I)) \geq c$, APPROX must return a value less than $f(p)$, and thus DECIDE outputs "$I \in S$." □

2.4. 1-IN-3-SAT. The NP-hard problem we use in our reductions is a variant of 3-SAT, the "monotone 1-in-3-SAT problem" [10, 23]. An instance I of monotone 1-in-3-SAT consists of a set of variables $V = \{v_1, v_2, \dots, v_n\}$ and a nonempty collection of clauses $\{c_t\}_{1 \leq t \leq s}$, each of size 3. (i.e., each c_t is a 3-element subset of V). For brevity, we henceforth omit the word "monotone," and refer to the problem as "1-in-3-SAT." $|I|$ denotes the *size* of the instance I according to some fixed encoding scheme. In particular, $|I|$ is always at least as large as the number of variables plus the number of clauses of instance I, and is not more than polynomially larger. A (*truth*) *assignment* is a function τ: $V \to \{0, 1\}$. An assignment τ *is a solution to* I if for every clause (v_x, v_y, v_z) of I, the multiset $\{\tau(v_x), \tau(v_y), (v_z)\} = \{0, 0, 1\}$. If we write that v_x is assigned true (respectively false) by τ, we mean $\tau(v_x) = 1$ (respectively, $\tau(v_x) = 0$). The decision problem for 1-in-3-SAT is to determine for any input instance I, whether or not there exists a solution to I.

2.5. COUNTER DFAS. The smallest consistent DFA for the examples of the reductions presented will be of the following special form. A *Counter DFA* (CDFA) over alphabet V is a deterministic finite automaton that counts the number of occurrences of characters in a subset V' of V mod p for some number p as follows: The labeled graph representing the CDFA consists of a simple cycle of p states, with the labeled edges of V' advancing one state around the cycle, and the labeled edges of $V - V'$ returning to the same state. Thus, each character of V' read increases the "count" by 1 mod p, and a character of $V - V'$ leaves the count unchanged. Further, the start state is the same as the unique final state; thus, CDFAs count from 0 to $p - 1$ mod p. CDFAs are a restricted subclass of DFAs that are contained in a class that is pac-learnable. Further discussion appears in Section 9.

Definition 2.11. Let τ: $V \to \{0, 1\}$ be a truth assignment to the variable set V. Then $C(p, \tau)$ is the CDFA that counts all true variables (i.e., counts the set $\tau^{-1}(1)$) mod p as described above.

3. *Forcing a Quadratic Gap*

Our main theorem will show that for any k, MIN-CON(DFA, NFA) is not opt^k-approximable. In this section, we first prove the special case for $k = 2$, that is, we show that no polynomial time approximation algorithm can guarantee less than a quadratic relationship between optimal and the solution it finds:

THEOREM 3.1. *If $P \neq NP$, then MIN-CON(DFA, NFA) is not opt^2-approximable.*

Besides motivating the general case in the next section, Theorem 3.1 is of independent interest for at least three reasons: (1) The proof involves interesting techniques that may be useful in other domains; (2) the proof gives some intuition for the polynomial nonapproximability results of the next section; and (3) the quadratic case provides a better nonapproximability result with respect to a number of the measures of approximability discussed in Section 9.

3.1. The Reduction

Proof of Theorem 3.1. By Lemma 2.10, Theorem 3.1 follows by exhibiting, for each of infinitely many positive integers p, a polynomial-time transformation R_p from instances of 1-in-3-SAT to MIN-CON(DFA, NFA) such that a quadratic gap is created, as required by Properties 2 and 3 of the hypothesis of Lemma 2.10. In particular, we describe for each odd number p, a reduction R_p, such that if I is any instance of a (monotone) 1-in-3-SAT problem, then $R_p(I)$ consists of two sets of strings, POS(p, I), and NEG(p, I), such that the following lemmas hold.

Lemma 3.2. *If assignment τ is a solution to I, then for any odd number p, $C(p, \tau)$ is a p-state DFA that is consistent with POS(p, I) and NEG(p, I), and no NFA (hence no DFA) with fewer states is consistent.*

Lemma 3.3. *If I has no solution, then for any odd number p, there does not exist an NFA with fewer than p^2 states that is consistent with POS(p, I) and NEG(p, I).*

Rather than describing R_p, we simply give the sets POS(p, I) and NEG(p, I) that are produced by R_p on input instance I. It is easily verified that R_p is computable in time polynomial in the size of I for any fixed odd number p. Section 3.2 is devoted to the proof of Lemma 3.2. In Section 3.3, a number of propositions are given that culminate with the proof of Lemma 3.3. These lemmas, together with Lemma 2.10, prove Theorem 3.1. □

If I is an instance of 1-in-3-SAT with variable set $V = \{v_1, v_2, \dots, v_n\}$, then the alphabet Σ over which POS(p, I) and NEG(p, I) are defined is given by $\Sigma = V \cup \{r\}$, where r is a symbol not appearing in V. (Here, and in Section 4, the alphabets used in our reductions depend on the input instance. In Section 6, we show how the reductions may be modified to work even when the alphabet is fixed as $\{0, 1\}$.) We need the following notation.

We use "$\equiv_p$" to denote congruence mod p. The function $[\cdot]_p$ is defined by $[a]_p = e$ iff e is the unique number such that $a \equiv_p e$, and such that $0 \le e \le p - 1$. Throughout all of Section 3 (and nowhere else), the variables $a, b, c, d, e, f, g, h, i, j, k, x, y, z$ are always in the following ranges:

- $0 \le a, b, c, d \le p^2$,
- $0 \le e, f, g, h \le p - 1$,
- $1 \le i, j, k \le p - 1$,
- $1 \le x, y, z \le n$.

Note. If i is in the range $1 \le i \le p - 1$, then $p - i$ is in the same range, and thus $-i \not\equiv_p 0$.

Definition 3.4. Let $p \geq 3$ be odd. Then define

- $q = (\prod_{x=1}^{n}(v_x)^p)r$. Thus, q is the concatenation of p copies of each variable, followed by the additional symbol r.
- For all x, $\alpha_x = \prod_{y=1}^{x-1}(v_y)^p$, and $\beta_x = (\prod_{y=x+1}^{n}(v_y)^p)r$. Thus, α_x is the unique prefix of q, and β_x is the unique suffix of q such that $\alpha_x(v_x)^p\beta_x = q$.
- For all x and i, $w_{x,i} = \alpha_x(v_x)^i\beta_x$. Thus, $w_{x,i}$ is a variant of q in which the substring $(v_x)^p$ has been replaced with $(v_x)^i$.

Before we describe the examples in detail, we give a description of the counter DFA that is constructed from a solution of the instance of 1-in-3-SAT. We do this to motivate the definition of the examples. After the examples are given, we show that the described machine is consistent with the examples.

Let I be any instance of a 1-in-3-SAT problem, and $\tau: V = \{v_1, v_2, \dots, v_n\} \to \{0, 1\}$ be an assignment with exactly one true variable per clause. Extend τ to domain $V \cup \{r\}$ by assigning $\tau(r) = 1$, and define $C(p, \tau)$ with respect to the (extended) assignment τ as in Definition 2.11. If w is any string, let $T(w)$ denote the total number of occurrences of variables assigned true, that is, variables v such that $\tau(v) = 1$. For any number s, we'll say *w advances by s* iff $T(w) \equiv_p s$. If w advances by s, then on reading w, $C(p, \tau)$ ends up in the $[s]_p$th state of the p-state cycle of $C(p, \tau)$. Thus, $C(p, \tau)$ accepts w if $s \equiv_p 0$, and $C(p, \tau)$ rejects w if $s \not\equiv_p 0$.

PROPOSITION 3.5. *Considering $C(p, \tau)$, it is easily verified that*

- *for all x, v_x advances by 1 if $\tau(v_x) = 1$, and v_x advances by 0 if $\tau(v_x) = 0$;*
- *r advances by 1, and for all numbers a, q^a advances by a;*
- *for all i and x, $w_{x,i}$ advances by $i + 1$ if $\tau(v_x) = 1$, and $w_{x,i}$ advances by 1 if $\tau(v_x) = 0$.*

3.1.1. *Motivation for the examples.* The string $w_{x,i}$ contains a single occurrence of r, all variables other than v_x occur a multiple of p times, and v_x occurs i times. Thus, in the CDFA $C(p, \tau)$, $w_{x,i}$ either advances by $i + 1$ or by 1 depending on whether τ assigns v_x true or false, respectively.

The examples are constructed to force the same properties in any consistent NFA of less than p^2 states. In particular, we force a loop in the NFA, and assign true all variables v_x such that for some i between 1 and $p - 1$, the string $w_{x,i}$ advances $i + 1$ around the loop, and false to those variables v_x such that for some j between 1 and $p - 1$, the string $w_{x,j}$ advances by 1 around the loop. Negative examples are used to enforce that $w_{x,i}$ cannot advance around the loop by any value other than 1 or $i + 1$. Additional negative examples enforce consistency, that is, that for no variable does there exist i and j such that from some state s, $w_{x,i}$ advances by $i + 1$, whereas from some other state t, $w_{x,j}$ advances by only 1. Further examples enforce that among those variables appearing in some clause of I, exactly one variable will be assigned true.

If for some i between 1 and $p - 1$, we have that $w_{x,i}$ advances by $i + 1$, we say that v_x "crosses," and if $w_{x,i}$ advances by only 1, we say that v_x "loops." The exact definitions of crossing and looping will be given later.

For each variable v_x, if for some $w_{x,i}$ it is the case that $w_{x,i}$ leaves some state of the loop and then returns to the loop, then the definitions and examples ensure that v_x is either looping or crossing, and thus v_x is assigned either true or false. If all variables are assigned, the examples enforce that the assignment is a solution to the 1-in-3-SAT instance. Thus, to avoid providing a solution, some variable must not be assigned. If variable v_x is unassigned, then we must have that starting from any state on the loop, each of the strings $w_{x,1}, \ldots, w_{x,p-1}$ do not return to the loop. It can be shown that all of the states these strings lead to are distinct. This forces $p - 1$ new states for each state on the loop. Since the loop has p states, we force a total of at least p^2 states.

In the following, we describe the examples. The sentence in parentheses following the name of each subset of examples suggests the function of the subset.

$\mathrm{POS}(p, I) = \{q^{p^2}\}$ (This example ensures that any NFA with less than p^2 states that accepts this example must have a loop).

$\mathrm{NEG}(p, I) = \mathrm{N1} \cup \mathrm{N2} \cup \mathrm{N3} \cup \mathrm{N4} \cup \mathrm{N5}$, where N1–N5 are defined below.

N1 (Any NFA consistent with $\mathrm{POS}(p, I)$ must have at least p states.)
For all a such that $a \not\equiv_p 0$, $q^a \in \mathrm{N1}$.

N2 ($w_{x,i}$ is only allowed to advance by $i + 1$ or by 1.)
For all a, x, i, b such that $b \not\equiv_p -(a + 1)$ and $b \not\equiv_p -(a + i + 1)$, $q^a w_{x,i} q^b \in \mathrm{N2}$.

N3 (v_x cannot cross and loop at the same time.)
For all a, x, i, b, j, c such that $c \equiv_p -(a + i + b + 2)$, $q^a w_{x,i} q^b w_{x,j} q^c \in$ N3.

N4 (No two variables of the same clause can cross.)
For all a, x, i, b, c, y, j, d such that there exists a clause containing the two variables v_x and v_y,

(a) If $d \equiv_p -(a + i + b + j + 2)$ and $(i + j) \not\equiv_p 0$, then $q^a w_{x,i} q^b w_{y,j} q^d \in \mathrm{N4}$.

(b) If $d \equiv_p -(a + b + c + 2i + j + 3)$ and $(i + j) \equiv_p 0$, then $q^a w_{x,i} q^b w_{x,i} q^c w_{y,j} q^d \in \mathrm{N4}$.

N5 (Not all three variables of a clause can loop.)
For all $a, x, i, b, y, j, c, z, k, d$ such that there exists a clause (v_x, v_y, v_z) and $d \equiv_p -(a + b + c + 3)$, $q^a w_{x,i} q^b w_{y,j} q^c w_{z,k} q^d \in \mathrm{N5}$.

PROPOSITION 3.6. *The number of examples in $\mathrm{POS}(p, I) \cup \mathrm{NEG}(p, I)$ is $O(p^{11}n^3)$, and the total number of characters (over alphabet $V \cup \{r\}$) in all examples of $\mathrm{POS}(p, I) \cup \mathrm{NEG}(p, I)$ is $O(p^{14}n^4)$.*

PROOF. $\mathrm{POS}(p, I)$ has only one element. From the respective ranges of a, b, c, d, i, j, k, x, y, and z, we immediately have that the total number of elements of: N1 is at most p^2; N2 is at most $pn(p^2 + 1)^2$; N3 is at most $p^2n(p^2 + 1)^3$; N4(a) is at most $p^2n^2(p^2 + 1)^3$; N4(b) is at most $p^2n^2(p^2 + 1)^4$; and N5 is at most $p^3n^3(p^2 + 1)^4$. Thus, the total number of elements of $\mathrm{POS}(p, I) \cup \mathrm{NEG}(p, I)$ is $O(p^{11}n^3)$. The longest example is of type N5, and has length $O(p^3n)$; thus, the total number of characters in all of the examples is $O(p^{14}n^4)$. □

3.2. A SMALL CONSISTENT COUNTER MACHINE. To prove Lemma 3.2, we show that for each assignment τ that is a solution to the instance I of 1-in-3-SAT, the corresponding p state CDFA $C(p, \tau)$ (defined above) is consistent with POS(p, I) and NEG(p, I), and that no NFA with fewer states is possible.

PROOF OF LEMMA 3.2. We first show consistency of $C(p, \tau)$ with each set of examples.

Consistency with POS(p, I). q^{p^2} advances by p^2, and is thus accepted since $p^2 \equiv_p 0$.

*Consistency with N*1. q^a advances by a, and thus is rejected if $a \not\equiv_p 0$.

*Consistency with N*2. Let a, x, i, b be such that $b \not\equiv_p -(a+1)$ and $b \not\equiv_p -(a+i+1)$. We show that the negative example $u = q^a w_{x,i} q^b$ is rejected by $C(p, \tau)$.

If $\tau(v_x) = 0$, then u advances by $a + 1 + b$, and since $b \not\equiv_p -(a+1)$, $C(p, \tau)$ rejects u.

If $\tau(v_x) = 1$, then u advances by $a + i + 1 + b$, and since $b \not\equiv_p -(a + i + 1)$, $C(p, \tau)$ rejects u.

*Consistency with N*3. Let a, x, i, b, j, c be such that $c \equiv_p -(a + i + b + 2)$; we show that the negative example $u = q^a w_{x,i} q^b w_{x,j} q^c$ is rejected by $C(p, \tau)$.

If $\tau(v_x) = 0$, then u advances by $a + 1 + b + 1 + c \equiv_p -i \not\equiv_p 0$, and is therefore rejected by $C(p, \tau)$. (Recall $1 \le i \le p - 1$.)

If $\tau(v_x) = 1$, then u advances by $a + i + 1 + b + j + 1 + c \equiv_p j \not\equiv_p 0$, and is also rejected by $C(p, \tau)$.

*Consistency with N*4(*a*). Let a, x, i, b, y, j, d be such that $d \equiv_p -(a + i + b + j + 2)$, $(i + j) \not\equiv_p 0$, and such that v_x and v_y appear together in some clause. We show that $C(p, \tau)$ rejects the negative example $u = q^a w_{x,i} q^b w_{y,j} q^d$. Note that since v_x and v_y appear in some clause together, we cannot have $\tau(v_x) = \tau(v_y) = 1$.

If $\tau(v_x) = 1$ and $\tau(v_y) = 0$, then u advances by $a + i + 1 + b + 1 + d \equiv_p -j \not\equiv_p 0$ and is rejected by $C(p, \tau)$.

If $\tau(v_x) = 0$ and $\tau(v_y) = 1$, then u advances by $a + 1 + b + j + 1 + d \equiv_p -i \not\equiv_p 0$, and is rejected by $C(p, \tau)$.

If $\tau(v_x) = \tau(v_y) = 0$, then u advances by $a + 1 + b + 1 + d \equiv_p -i - j \not\equiv_p 0$ (since in this subcase we have $i + j \not\equiv_p 0$) and is rejected by $C(p, \tau)$.

*Consistency with N*4(*b*). Let a, x, i, b, c, y, j, d be such that $d \equiv_p -(a + b + c + 2i + j + 3)$, $(i + j) \equiv_p 0$, and such that v_x and v_y appear together in some clause. We show that $C(p, \tau)$ rejects the negative example $u = q^a w_{x,i} q^b w_{x,i} q^c w_{y,j} q^d$.

If $\tau(v_x) = 1$ and $\tau(v_y) = 0$, then u advances by $a + i + 1 + b + i + 1 + c + 1 + d \equiv_p -j \not\equiv_p 0$ and thus is rejected by $C(p, \tau)$.

If $\tau(v_x) = 0$ and $\tau(v_y) = 1$, then u advances by $a + 1 + b + 1 + c + j + 1 + d \equiv_p -2i$. Thus, u is accepted by $C(p, \tau)$ iff $2i \equiv_p 0$. But since p is odd, this implies that $i \equiv_p 0$. But recall $1 \le i \le p - 1$, so $-2i \not\equiv_p 0$ and u is rejected by $C(p, \tau)$.

If $\tau(v_x) = \tau(v_y) = 0$, then u advances by $a + 1 + b + 1 + c + 1 + d \equiv_p -2i - j \equiv_p -i - (i + j) \equiv_p -i$ (since $i + j \equiv_p 0$). And since $-i \not\equiv_p 0$, u is rejected by $C(p, \tau)$.

Consistency with N5. Let $a, x, i, b, y, j, c, z, k$ be such that $d \equiv_p -(a + b + c + 3)$, and such that there exists a clause (v_x, v_y, v_z). Then $u = q^a w_{x,i} q^b w_{y,j} q^c w_{z,k} q^d$ is a negative example. Exactly one of $\{\tau(v_x), \tau(v_y), \tau(v_z)\}$ is true; thus, u advances by either i, j, or k, none of which are congruent to 0 mod p, and thus $C(p, \tau)$ rejects u.

Thus, $C(p, \tau)$, with p states, is consistent with POS(p, I) and NEG(p, I).

To complete the proof of Lemma 3.2, we still need to show that any consistent NFA must have at least p states. Let $A = (Q, \Sigma, \delta, s_{\text{init}}, F)$ be an NFA that is consistent with POS(p, I) and NEG(p, I). Using the graph representation of A, since the string $q^{p^2} \in$ POS(p, I), it is accepted by A, and thus there is a path ψ (labeled with q^{p^2}) in A from s_{init} to some accepting state $s_+ \in F$. For any w that is a prefix of q^{p^2}, we say *w leads along ψ to state t* if the initial segment of ψ labeled with w leads from s_{init} to t. For each a $(0 \leq a \leq p^2)$, let s_a denote the state that q^a leads to along ψ. It suffices to show that the states $\{s_e\}_{0 \leq e \leq p-1}$, are distinct. If $s_f = s_g$ for $0 \leq f < g \leq p - 1$, then $q^{p^2-(g-f)}$ is accepted. But this contradicts the consistency with the examples of N1, since $p^2 - (g - f) \not\equiv_p 0$. This completes the proof of Lemma 3.2. □

3.3. FORCING QUADRATICALLY LARGER NFAS. To prove Lemma 3.3, we must show, for any instance I of 1-in-3-SAT that has no solution, that no NFA with strictly less than p^2 states is consistent with the examples POS(p, I) and NEG(p, I). We prove the contrapositive, namely, that if $A = (Q, \Sigma, \delta, s_{\text{init}}, F)$ is an NFA that is consistent with POS(p, I) and NEG(p, I) and such that $|Q| < p^2$, then I has a solution. The proof is presented following the introduction and proof of a number of supporting propositions.

As in the proof of Lemma 3.2, we use the graph representation of A, and recall that the string $q^{p^2} \in$ POS(p, I) defines the path ψ from s_{init} to some accepting state $s_+ \in F$, and that for each a, s_a is the unique state that q^a leads to along ψ.

PROPOSITION 3.7. *There exists a, b, with $0 \leq a < b \leq p^2$, such that $s_a = s_b$.*

PROOF. If not, then $|Q| > p^2$. □

Let a_0 be the smallest number such that for some $b > a_0$, $s_{a_0} = s_b$. Let $b_0 > a_0$ be the smallest number such that $s_{a_0} = s_{b_0}$. Thus, following powers of q, the path ψ in A labeled with q^{p^2} leading to s_+ contains a loop as shown in Figure 1. Let L be the states on the loop that are reachable by powers of q, that is, $L = \{s_{a_0}, s_{a_0+1}, \ldots, s_{b_0-1}\}$.

PROPOSITION 3.8. $|L| > 0$ *and* $|L| \equiv_p 0$.

PROOF. Since $s_{a_0} \in L$, $1 \leq |L| \leq |Q| < p^2$. To complete the proof, we show $|L| \equiv_p 0$. Observe that q^{a_0} leads to s_{a_0}, and $q^{p^2-b_0}$ leads from $s_{a_0} = s_{b_0}$ to s_+. Thus $u = q^{a_0+p^2-b_0} = q^{p^2-(b_0-a_0)}$ leads to s_+ and is accepted by A. But if $b_0 - a_0 \not\equiv_p 0$, then u is a negative example (N1). Therefore, $b_0 - a_0 \equiv_p 0$, and since $|L| = b_0 - a_0$, $|L| \equiv_p 0$. □

For each e, define $S_e = \{s_a\colon a_0 \leq a \leq b_0 - 1 \text{ and } [a]_p = e\}$, that is, S_e is the set of states on L that are reachable along ψ by some power of a between a_0 and $b_0 - 1$ that is congruent to e, where $0 \leq e \leq p - 1$.

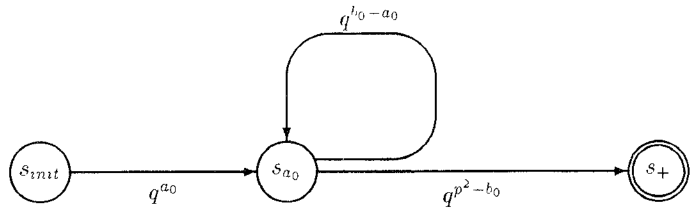

FIG. 1. The loop of path ψ in A.

PROPOSITION 3.9 (PREFIX PROPERTY). *For all e and for all $s \in S_e$, there exists a such that $[a]_p = e$, q^a leads to s, and $a_0 \leq a \leq b_0 - 1$.*

PROOF. By definition. □

PROPOSITION 3.10. *For all e, S_e is nonempty, and in particular, $s_{a_0+[e-a_0]_p} \in S_e$.*

PROOF. Since $|L| \geq p$, $a_0 + [e - a_0]_p < b_0$, and the proposition follows from the definition of S_e. □

PROPOSITION 3.11 (REACHABILITY PROPERTY). *If $s_a \in S_e$ and $s_b \in S_f$, then there exists $c \equiv_p f - e$ such that q^c leads from s_a to s_b.*

PROOF. If $a_0 \leq a \leq b \leq b_0$, then q^{b-a} leads from s_a to s_b. Let $c = b - a$, observing that indeed, $0 \leq c \leq p^2$, and $c \equiv_p b - a \equiv_p f - e$.

If $a_0 \leq b \leq a \leq b_0$, then q^{b_0-a} leads from s_a to $s_{b_0} = s_{a_0}$. Then q^{b-a_0} leads from s_{a_0} to s_b; thus, $q^{b_0-a+b-a_0}$ leads from s_a to s_b. Now let $c = b_0 - a + b - a_0 = (b_0 - a_0) - (a - b)$. Clearly, $0 \leq a - b \leq b_0 - a_0 \leq p^2$, so c is at most p^2. Finally, $c \equiv_p b - a \equiv_p f - e$. (Recall $b_0 - a_0 \equiv_p 0$.) □

PROPOSITION 3.12 (SUFFIX PROPERTY). *For all a, if $s_a \in S_e$, then there exists $b \equiv_p -e$ such that q^b leads from s_a to s_+.*

PROOF. Recall that the path ψ is labeled with q^{p^2}. Then q^a leads to s_a and q^b leads from s_a to s_+ when $b = p^2 - a$. Since $s_a \in S_e$, $a \equiv_p e$ and thus $b \equiv_p -e$. □

PROPOSITION 3.13. *For all e, f, if $e \neq f$, then $S_e \cap S_f = \emptyset$.*

PROOF. Assume that $e \neq f$ and that $t \in S_e \cap S_f$. By the Prefix Property, there exists a such that $a \equiv_p e$ such that q^a leads to t. By the Suffix Property, there exists $b \equiv_p -f$, such that q^b leads from t to s_+. Then $q^a q^b$ leads to s_+, and is accepted by A. But $a + b \equiv_p f - e \not\equiv_p 0$ (since e and f are between 0 and $p - 1$ and are distinct), and thus $q^a q^b$ is a negative example of type N1, contradicting the consistency of A. □

For each e, we use "t_e" to denote any element of S_e. We now define, for each variable v_x, what it means for v_x to be *crossing*, *looping*, or to be *off*.

Crossing: v_x *crosses* if there exists e, i, $t_e \in S_e$ and $t_{[e+i+1]_p} \in S_{[e+i+1]_p}$ such that $w_{x,i}$ leads from t_e to $t_{[e+i+1]_p}$.

Looping: v_x *loops* if there exists e, i, $t_e \in S_e$, and $t_{[e+1]_p} \in S_{[e+1]_p}$ such that $w_{x,i}$ leads from t_e to $t_{[e+1]_p}$.

Off: v_x is *off* if v_x neither crosses nor loops.

PROPOSITION 3.14. *For all x, i, e, f, $t_e \in S_e$, and $t_f \in S_f$, if $w_{x,i}$ leads from t_e to t_f, then $f \equiv_p e+1$ or $f \equiv_p e+i+1$. Hence, if $w_{x,i}$ leads from some S_e to some S_f, then v_x is not off.*

PROOF. Let e, f, i, x, t_e, t_f be as in the hypothesis of the proposition. Then,

$$\begin{array}{llllll}
\exists a & \text{such that } q^a & \text{leads from } s_{\text{init}} & \text{to } t_e & \text{and} & a \equiv_p e, \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_f, & & \\
\exists b & \text{such that } q^b & \text{leads from } t_f & \text{to } s_+ & \text{and} & b \equiv_p -f.
\end{array}$$

Thus, $u = q^a w_{x,i} q^b$ is accepted by A. By examples (N2), unless $b \equiv_p -(a+1)$ or $b \equiv_p -(a+i+1)$, u is a negative example. Since A is consistent with the examples and u is accepted, either $b \equiv_p -(a+1)$ or $b \equiv_p -(ai{+}{+}\,1)$, and thus $f \equiv_p e+1$ or $f \equiv_p e+i+1$. □

PROPOSITION 3.15. *For all x, either v_x does not cross, or v_x does not loop.*

PROOF. Suppose that v_x crosses and v_x loops. Then there exists e, f, i, j, and $t_e, t_{[e+i+1]_p}, t_f, t_{[f+1]_p}$ (elements of S_e, $S_{[e+i+1]_p}$, S_f, $S_{[f+1]_p}$, respectively) such that $w_{x,i}$ leads from t_e to $t_{[e+i+1]_p}$, and $w_{x,j}$ leads from t_f to $t_{[f+1]_p}$. Furthermore,

$$\begin{array}{llllll}
\exists a & \text{such that } q^a & \text{leads from } s_{\text{init}} & \text{to } t_e & \text{and } a \equiv_p e \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_{[e+i+1]_p} & \\
\exists b & \text{such that } q^b & \text{leads from } t_{[e+i+1]_p} & \text{to } t_f & \text{and } b \equiv_p f-(e+i+1) \\
 & w_{x,j} & \text{leads from } t_f & \text{to } t_{[f+1]_p} & \\
\exists c & \text{such that } q^c & \text{leads from } t_{[f+1]_p} & \text{to } s_+ & \text{and } c \equiv_p -(f+1).
\end{array}$$

Thus, $u = q^a w_{x,i} q^b w_{x,j} q^c$ leads to s_+, and is accepted by A. Notice that

$$-(f+1) = -(e+i+(f-(e+i+1))+2),$$

and thus

$$c \equiv_p -(a+i+b+2)$$

and u is a negative example (type N3) accepted by A, a contradiction. □

PROPOSITION 3.16. *For all x, y, if there exists a clause of I containing both v_x and v_y, then at least one of v_x and v_y does not cross.*

PROOF. Suppose that v_x and v_y both cross and that they occur in the same clause. Then there exists e, f, i, j, and $t_e, t_{[e+i+1]_p}, t_f, t_{[f+j+1]_p}$ (elements of S_e, $S_{[e+i+1]_p}$, S_f, $S_{[f+j+1]_p}$, respectively) such that $w_{x,i}$ leads from t_e to $t_{[e+i+1]_p}$ and $w_{y,j}$ leads from t_f to $t_{[f+j+1]_p}$.

Case 1. $i + j \not\equiv_p 0$. Then

$$\begin{array}{llllll}
\exists a & \text{such that } q^a & \text{leads from } s_{\text{init}} & \text{to } t_e & \text{and } a \equiv_p e \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_{[e+i+1]_p} & \\
\exists b & \text{such that } q^b & \text{leads from } t_{[e+i+1]_p} & \text{to } t_f & \text{and } b \equiv_p f-(e+i+1) \\
 & w_{y,j} & \text{leads from } t_f & \text{to } t_{[f+j+1]_p} & \\
\exists d & \text{such that } q^d & \text{leads from } t_{[f+j+1]_p} & \text{to } s_+ & \text{and } d \equiv_p -(f+j+1).
\end{array}$$

Then $q^a w_{x,i} q^b w_{y,j} q^d$ leads to s_+ and is accepted by A. Further, $d \equiv_p -(a+i+b+j+2)$ and $i+j \not\equiv_p 0$, so it is a negative example of type N4(a), a contradiction.

Case 2. $i + j \equiv_p 0$. Then

$$\begin{array}{llllll}
\exists a & \text{such that } q^a & \text{leads from } s_{\text{init}} & \text{to } t_e & \text{and } a \equiv_p e \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_{[e+i+1]_p} & \\
\exists b & \text{such that } q^b & \text{leads from } t_{[e+i+1]_p} & \text{to } t_e & \text{and } b \equiv_p e-(e+i+1) \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_{[e+i+1]_p} & \\
\exists c & \text{such that } q^c & \text{leads from } t_{[e+i+1]_p} & \text{to } t_f & \text{and } c \equiv_p f-(e+i+1) \\
 & w_{y,j} & \text{leads from } t_f & \text{to } t_{[f+j+1]_p} & \\
\exists d & \text{such that } q^d & \text{leads from } t_{[f+j+1]_p} & \text{to } s_+ & \text{and } d \equiv_p -(f+j+1).
\end{array}$$

Then $q^a w_{x,i} q^b w_{x,i} q^c w_{y,j} q^d$ leads to s_+ and is accepted by A. Further, $d \equiv_p -(a+b+c+2i+j+3)$ and $i+j \equiv_p 0$, so it is a negative example of type N4(b), a contradiction. □

PROPOSITION 3.17. *For all* x, y, z, *if* (v_x, v_y, v_z) *is a clause of* I, *then not all of* $\{v_x, v_y, v_z\}$ *loop.*

PROOF. Suppose for some clause (v_x, v_y, v_z), each of $\{v_x, v_y, v_z\}$ loops. Then there exists e, f, g, i, j, k, and $t_e, t_{[e+1]_p}, t_f, t_{[f+1]_p}, t_g, t_{[g+1]_p}$ (elements of S_e, $S_{[e+1]_p}$, S_f, $S_{[f+1]_p}$, S_g, $S_{[g+1]_p}$, respectively) such that $w_{x,i}$ leads from t_e to $t_{[e+1]_p}$, $w_{y,j}$ leads from t_f to $t_{[f+1]_p}$, $w_{z,k}$ leads from t_g to $t_{[g+1]_p}$. Further,

$$\begin{array}{llllll}
\exists a & \text{such that } q^a & \text{leads from } s_{\text{init}} & \text{to } t_e & \text{and } a \equiv_p e \\
 & w_{x,i} & \text{leads from } t_e & \text{to } t_{[e+1]_p} & \\
\exists b & \text{such that } q^b & \text{leads from } t_{[e+1]_p} & \text{to } t_f & \text{and } b \equiv_p f-(e+1) \\
 & w_{y,j} & \text{leads from } t_f & \text{to } t_{[f+1]_p} & \\
\exists c & \text{such that } q^c & \text{leads from } t_{[f+1]_p} & \text{to } t_g & \text{and } c \equiv_p g-(f+1) \\
 & w_{z,k} & \text{leads from } t_g & \text{to } t_{[g+1]_p} & \\
\exists d & \text{such that } q^d & \text{leads from } t_{[g+1]_p} & \text{to } s_+ & \text{and } d \equiv_p -(g+1).
\end{array}$$

Then $q^a w_{x,i} q^b w_{y,j} q^c w_{z,k} q^d$ leads to s_+ and is accepted by A. Further, $d \equiv_p -(a+b+c+3)$, so it is a negative example of type N5, a contradiction. □

PROPOSITION 3.18. *For all* x, v_x *is not off.*

PROOF. We show that if for some x, v_x was off, then A would have at least p^2 states, a contradiction. Suppose v_x is off. Let $0 \le e, f, g, h \le p-1$. Define t_e to be the state $s_{a_0+[e-a_0]_p}$ (the state that $q^{a_0+[e-a_0]_p}$ leads to along ψ). Similarly, define t_f to be the state $s_{a_0+[f-a_0]_p}$. By Proposition 3.10, $t_e \in S_e$ and $t_f \in S_f$. To show that A has at least p^2 states, it suffices to show:

CLAIM 3.19. *If $\langle e, g\rangle \neq \langle f, h\rangle$, then the state that $q^{a_0+[e-a_0]_p}\alpha_x(v_x)^g$ leads to along ψ (denote this state by $t_e(\alpha_x(v_x)^g)$) is distinct from the state that $q^{a_0+[f-a_0]_p}\alpha_x(v_x)^h$ leads to along ψ (denoted $t_f(\alpha_x(v_x)^h)$).*

Note that the state $t_e(\alpha_x(v_x)^g)$ is well defined, since $q^{a_0+[e-a_0]_p}\alpha_x(v_x)^g$ is a prefix of q^{p^2}. Similarly, the state $t_f(\alpha_x(v_x)^h)$ is well defined. We prove this claim in two cases:

Case 1. $g = h$ and $e \neq f$.

Suppose $t_e(\alpha_x(v_x)^g) = t_f(\alpha_x(v_x)^h) = s$. Then let $u = (v_x)^{p-g}\beta_x = (v_x)^{p-h}\beta_x$. Then since $\alpha_x(v_x)^g u = \alpha_x(v_x)^h u = q$, u leads from s to a state t that must be in $S_{[e+1]_p} \cap S_{[f+1]_p}$. This is a contradiction, because $e \neq f$, thus $[e+1]_p \neq [f+1]_p$, and by Proposition 3.13, $S_{[e+1]_p} \cap S_{[f+1]_p} = \emptyset$.

Case 2. $g \neq h$.

Without loss of generality, $g > h$. Let $t_e(\alpha_x(v_x)^g) = t_f(\alpha_x(v_x)^h) = s$. Since $\alpha_x(v_x)^g(v_x)^{p-g}\beta_x = q$, $(v_x)^{p-g}\beta_x$ leads from s to an element of $S_{[e+1]_p}$. But $t_f(\alpha_x(v_x)^h) = s$ also, so $u = \alpha_x(v_x)^h(v_x)^{p-g}\beta_x$ leads from t_f to an element of $S_{[e+1]_p}$. Notice that $1 \leq h + p - g \leq p - 1$ because $0 \leq h < g \leq p - 1$. Thus, for some $1 \leq i \leq p - 1$, $u = w_{x,i}$, and leads from an element of S_f to an element of $S_{[e+1]_p}$. By Proposition 3.14, v_x is either looping or crossing, contradicting the assumption that v_x is off. This completes the proof of Claim 3.19 and Proposition 3.18. □

PROOF OF LEMMA 3.3. We now show that (since A has fewer than p^2 states, and is consistent with POS(p, I) and NEG(p, I)) there is solution for I. By Proposition 3.18, for all x, v_x is not off, and hence either crosses or loops. Consider the assignment τ: $V \to \{0, 1\}$ defined by $\tau(v_x) = 1$ if v_x crosses, and $\tau(v_x) = 0$ if v_x loops. By Proposition 3.15, v_x cannot both cross and loop, and thus τ is well defined. Further, by Proposition 3.16, within any clause of I, at most one variable is assigned true. Finally, by Proposition 3.17, for each clause, not all variables can be assigned false. Thus, τ is a solution for I. This completes the proof of Lemma 3.3, which together with Lemmas 3.2 and 2.10, completes the proof of Theorem 3.1. □

4. *Forcing a Polynomial Gap*

In this section, we extend the quadratic gap of the previous section to a polynomial of arbitrary degree, obtaining a stronger nonapproximability result for the minimum consistent DFA problem over an arbitrary alphabet.

THEOREM 4.1. *For all positive integers k, MIN-CON(DFA, NFA) is not opt^k-approximable unless $P = NP$.*

PROOF. As in the quadratic case, we provide a reduction from the 1-in-3-SAT problem to the MIN-CON(DFA, NFA) problem that introduces a gap, but this time between p and p^k, instead of p and p^2. Let k be any constant, and let $m = 3 \cdot 2^{k-1}$. We show that for all sufficiently large primes p (in particular, for $p > 2^{k-1+m}$), there exists a reduction $R_{p,k}$ that is computable in polynomial time (Proposition 4.10 below) that takes as input an instance I of a 1-in-3-SAT problem, and produces two sets POS(p, k, I) and NEG(p, k, I) that satisfy Lemmas 4.2, 4.3, and 4.4 (stated immediately below). It follows that the hypothesis of Lemma 2.10 is satisfied with $f(opt) = opt^k$, and thus unless P = NP, MIN-CON(DFA, NFA) is not opt^k-approximable. □

The key lemmas used are analogs of Lemmas 3.2 and 3.3, showing that there is a p state counter machine $C(p, \tau)$ (Lemma 4.2), which is in fact optimal (Lemma 4.3), and that any NFA with fewer than opt^k states provides a solution to a 1-in-3-SAT problem (Lemma 4.4).

LEMMA 4.2. *Let I be an instance of 1-in-3-SAT. If τ is a solution of I, then for all positive integers k and p, $C(p, \tau)$ is consistent with $POS(p, k, I)$ and $NEG(p, k, I)$. Thus, if I has some solution, then there exists a consistent p state DFA.*

LEMMA 4.3. *Let k and p be any positive integers, and let I be any instance of 1-in-3-SAT. Then any NFA that is consistent with $POS(p, k, I)$ and $NEG(p, k, I)$ has at least p states.*

LEMMA 4.4. *Let k be any positive integer, and let p be a prime such that $p > 2^{k-1+m}$. If I is any instance of 1-in-3-SAT, and if $A = (Q, \Sigma, \delta, s_{init}, F)$ is an NFA such that $|Q| < p^k$ and such that A is consistent with $POS(p, k, I)$ and $NEG(p, k, I)$, then I has some solution.*

Lemmas 4.2 and 4.3 are proved in Section 4.3, and Lemma 4.4 is proved in Section 4.4. Before delving into the details of the proofs, we discuss how the proof of Theorem 3.1 suggests the approach taken to prove the more general result.

We begin with the same underlying principle: If assignment τ is a solution of the 1-in-3-SAT formula, then we would like the counter machine $C(p, \tau)$ to be consistent with all positive and negative examples. Our goal is to construct a polynomial-sized set of examples that maintains this property, and such that if a small (less than p^k state) consistent NFA is provided, then a solution of the 1-in-3-SAT formula may be found.

Recall that in the motivation for the examples in the quadratic case, the string $w_{x,i}$ played a special role in helping determine whether v_x should be assigned true or false. Either $w_{x,i}$ advanced by $i + 1$ around the loop, or $w_{x,i}$ advanced by only one state. In the string $w_{x,i}$, variable v_x occurs a number of times congruent to i mod p; all other variables appear 0 times (mod p); and the special symbol r appears exactly once. Thus, another way of interpreting the fact that $w_{x,i}$ advances by $i + 1$ is to write that i occurrences of v_x plus one occurrence of r equals $i + 1$. More succinctly, "$i \cdot v_x + r \equiv_p i + 1$," where now we interpret v_x and r as a variables with values either 0 or 1. Since we know that $r = 1$ (i.e., r advances by 1), this reduces to $i \cdot v_x \equiv_p i$, and this equation tells us (if p is prime) that v_x must equal 1 (true).

The main idea in the proof of Theorem 4.1 is to extend this interpretation in the following way: Suppose some string containing i occurrences of v_x and j occurrences of v_y is found to advance around a loop a total of s states. Then we write the equation $i \cdot v_x + j \cdot v_y = s$. We construct examples such that in any consistent NFA with fewer than p^k states, for any k-tuple of variables we can find such an equation relating the k variables. By exploiting special properties of such systems of equations, we can construct negative examples such that the set of solutions must contain a solution of the 1-in-3-SAT instance.

As a final intuition before proceeding with technical details, we offer the following brief summary of the proof of Theorem 4.1. Note that a string is accepted by $C(p, \tau)$ iff the number of true variables (according to assignment τ) in the string is congruent to 0 mod p. In fact, if a string γ leads from any state in $C(p, \tau)$ back to that state, then the number of true variables in γ is congruent to 0. We may rewrite the above property of $C(p, \tau)$ as follows: For any string γ, Let $\vec{\gamma} = \langle x_1, x_2, \ldots, x_n \rangle$, such that for each $i, 1 \leq i \leq n$, the number of occurrences of v_i in γ is congruent to x_i mod p. Then γ leads around a cycle in $C(p, \tau)$ iff $\vec{\gamma} \cdot \tau = 0$, where the assignment τ is interpreted as a Boolean vector of length n, "$\cdot$" is the dot product, and all arithmetic is mod p. Consequently, given a collection of strings $\{\gamma_i\}$ that lead around a cycle in some unknown counter machine $C(p, \tau)$, one way of determining τ would be to solve the simultaneous system of equations $\{\vec{\gamma}_i \cdot \vec{x} = 0\}$. This suggests a strategy for constructing POS(p, k, I), NEG(p, k, I): Try to force the above property in *any* small consistent NFA, not simply a counter DFA. Thus, we construct POS(p, k, I) and NEG(p, k, I) such that

—if τ is any solution of I, then $C(p, \tau)$ (which has p states) is consistent with POS(p, k, I) and NEG(p, k, I);
—a single carefully constructed positive example forces a cycle in any accepting NFA with *strictly less than* p^k states;
—from the cycle, a set of strings $\{\gamma_i\}$ may be extracted;
—if S is the matrix with rows $\{\vec{\gamma}_i\}$ representing the equations $\{\vec{\gamma}_i \cdot \vec{x} = 0\}$, then the set of solutions to the system S contains an element that is $\{0, 1\}$ valued, and is a solution of I. Thus, the existence of a consistent NFA with less than p^k states implies that I has some solution.

The last property is achieved by including in NEG(p, k, I) examples that rule out consistent automata with less than p^k states whose induced set of equations (those extracted from strings leading around the cycle) do not include a solution of I.

4.1. More Definitions and Technical Lemmas. Let k and p be any positive integers. Throughout the rest of the paper, the constant m is defined by $m = 3 \cdot 2^{k-1}$. Let $P = \{0, 1, \ldots, p-1\}$. P^n denotes all vectors of length n with elements in P. Vectors $\vec{x}, \vec{y}, \vec{z}$ will always denote elements of P^n. For a row vector $\vec{x}$, we let $\vec{x}^T$ denote the transpose (column vector) of $\vec{x}$. All vectors are indexed starting at 1, thus $\vec{x} = \langle x_1, x_2, \ldots x_n \rangle$. We assume the standard lexicographic ordering on P^n. Thus, $\vec{x} < \vec{y}$ indicates that $\vec{x}$ comes first in the lexicographic order. Matrices are sets of row vectors in P^n. All operations involving vectors or matrices are mod p. The binary operator "$\cdot$" denotes the dot product (mod p), except where both operands are scalar values, in which case multiplication is denoted. $COL(M)$ denotes the set $\{i$: the ith column in M is nonzero$\}$. Note that $COL(M)$ is also defined if M consists of only one row. Let $K(M)$ denote the *kernel* of M, that is, $K(M) = \{\vec{x} : M\vec{x}^T = \vec{0}\}$, and let *span*$(M)$ be the set of all linear combinations of rows of M. Recall from linear algebra that if M is a matrix and B is a basis of M (more precisely, a basis of the vector space *span*(M)), then $K(M) = K(B)$.

Any assignment τ may also be interpreted as a vector $\langle \tau(v_1, \tau(v_2), \ldots, \tau(v_n) \rangle \in \{0, 1\}^n \subseteq P^n$. The symbol τ will be used to denote either the function, or the vector; the meaning will be clear from context. For example, in

"$\tau(v_x)$", the function is denoted, whereas the vector is denoted in "$\vec{x} \cdot \tau$." Similarly, any vector $\vec{x} \in \{0,1\}^n$ may be interpreted as a truth assignment, in which v_i is assigned false (respectively, true) iff $x_i = 0$ (respectively, $x_i = 1$).

Definition 4.5. Let M be a matrix of nonzero rows.

—A set $X \subseteq \{1, 2, \ldots n\}$ is *free* with respect to M if for every row $\vec{x}$ of M, $COL(\vec{x}) \not\subseteq X$.
—M is *k-closed* if for every $X \subseteq \{1, 2, \ldots, n\}$ such that $|X| = k$, X is not free with respect to M, that is, for each X of size k there exists a row $\vec{x}$ of M such that $COL(\vec{x}) \subseteq X$.
—A maximal free set C (with respect to a matrix M) is called a *core* of M.

Note that if M is a matrix of nonzero rows, then M has a core, since the empty set is free.

LEMMA 4.6. *Each core C of a k-closed matrix M has size at most $k - 1$, and for each number $i \in \{1, \ldots, n\} - C$, there is a row $\vec{x}$ of M (called a determining row of index i) such that $COL(\vec{x}) \subseteq C \cup \{i\}$ and such that x_i is nonzero.*

PROOF. Assume C is a core and has size at least k. Then, let C' be any subset of C of size k. Since M is k-closed, M contains a row $\vec{x}$ with $COL(\vec{x}) \subseteq C' \subseteq C$ and this contradicts the fact that C is free with respect to M.

To prove the second part of the lemma, assume that there is a number $i \in \{1, \ldots, n\} - C$ such that there does not exist a row $\vec{x}$ of M with $COL(\vec{x}) \subseteq C \cup \{i\}$ and such that $x_i \neq 0$. Then either (1) there does not exist a row $\vec{x}$ of M with $COL(\vec{x}) \subseteq C \cup \{i\}$, or else (2) there exists at least one row $\vec{x}$ of M with $COL(\vec{x}) \subseteq C \cup \{i\}$, such that $x_i = 0$. If (1) is the case, then $C \cup \{i\}$ is free with respect to M, contradicting the maximality of C. If (2) is the case, then $COL(\vec{x}) \subseteq C$, contradicting the fact that C is free. In either case, we have a contradiction; thus, the second part of the lemma must hold. □

LEMMA 4.7. *If p is prime, then any subset B of $span(M)$ of size larger than $|span(M)|/p$ contains a basis of M.*

PROOF. Let B be as above and let B' be any basis of M. If B does not contain a basis of M, then there is a row $\vec{x}$ of B' that is not in $span(B)$. Now observe that for any $r, r', \vec{y}, \vec{z}$ such that $0 \leq r, r' \leq p - 1, \vec{y}, \vec{z} \in span(B)$, and for which $\langle r, \vec{y} \rangle \neq \langle r', \vec{z} \rangle$, we have $r\vec{x} + \vec{y} \neq r'\vec{x} + \vec{z}$. To see this, observe that if $r = r'$ then $\vec{y} \neq \vec{z}$ and then trivially $r\vec{x} + \vec{y} \neq r'\vec{x} + \vec{z}$. On the other hand, if $r \neq r'$ and $r\vec{x} + \vec{y} = r'\vec{x} + \vec{z}$, then $\vec{x} = (\vec{z} - \vec{y})/(r - r') \in span(B)$, which is a contradiction. Note that division is well defined since p is prime. Thus, the set $\{r\vec{x} + \vec{y} : 0 \leq r \leq p - 1, \vec{y} \in B\}$ has size $p|B| > |span(M)|$. But this set is clearly a subset of $span(M)$, and we have a contradiction. Thus, B must contain a basis of M. □

Definition 4.8. If I is an instance of 1-in-3-SAT, then $SOL(I) = \{\vec{x} \in \{0,1\}^n : n$ is the number of variables of I and $\vec{x}$ is a solution to the instance $I\}$. If V is the set of variables of I and $V' \subseteq V$, then the instance *I restricted to V'* (written $I|_{V'}$) is the instance of 1-in-3-SAT over the same variable set V, that contains exactly the clauses c of I for which every element of c is in V'.

Note that the length of the vectors of $SOL(I|_{V'})$ is the number of variables in the original instance I.

4.2. THE REDUCTION AND EXAMPLES. Let I be an instance of 1-in-3-SAT, with variables $V = \{v_1, v_2, \dots, v_n\}$. We let the alphabet Σ for the problem MIN-CON(DFA, NFA) be V. In Section 6, we refine the reduction so that $\Sigma = \{0, 1\}$ regardless of the instance I.

Recall that for any string $\gamma \in \Sigma^*$, $\vec{\gamma} = \langle x_1, x_2, \dots x_n \rangle$, such that for each i, $1 \leq i \leq n$, the number of occurrences of v_i in γ is congruent to x_i mod p.

We now define a special word q, which is specified by a product (denoting concatenation) of many subwords. Since the product sign below denotes concatenation, to be unambiguous, we must specify the order in which the terms (subwords) are concatenated: The choice in the product below is made in lexicographic order of the vectors $\vec{x}$.

$$q = \prod_{\substack{\vec{x} \in P^n \\ |COL(\vec{x})| \leq k}} v_1^{x_1} v_2^{x_2} \cdots v_n^{x_n} v_1^{p - x_1} v_2^{p - x_2} \cdots v_n^{p - x_n}.$$

Note that $\vec{q} = \vec{0}$. Whenever they appear, α and β (and subscripted versions) will denote prefixes and suffixes, respectively, of q.

On input I, the transformation $R_{p,k}$ outputs $R_{p,k}(I)$ consisting of the two sets, POS(p, k, I) and NEG(p, k, I). We show that these sets have the properties claimed above. Note that the transformation $R_{p,k}$ need only be computable in time polynomial in $|I|$, since p and k are constants. However, in Section 6, we use the fact, proved below, that the transformation is also polynomial in the *value* p. (The dependence on k is doubly exponential however.) We now describe POS(p, k, I) and NEG(p, k, I).

POS(p, k, I) consists only of the word q^{p^k}.

NEG(p, k, I) is constructed as follows: Let $\{\alpha_i\}_{i=1}^m$ be any collection of m prefixes of q (recall $m = 3 \cdot 2^{k-1}$), let $\{\beta_i\}_{i=1}^m$, be any collection of m suffixes of q, and let $\langle p_1, p_2, \dots, p_m \rangle$ be any element of P^m. Define $\gamma_i = \alpha_i \beta_i$. Let $\gamma = \prod_{i=1}^m (\gamma_i)^{p_i}$. Then, if there exists a set D such that

(1) $|D| \leq k - 1 + m$,
(2) $\mathrm{COL}(\vec{\gamma}) \subseteq D \subseteq V$,
(3) $\mathrm{SOL}(I/_D) \cap K(\vec{\gamma}) = \varnothing$,

then for any numbers a, b, c $(0 \leq a, b, c \leq p^k)$, include in the set NEG$(p, k, I)$ the string $\gamma_{a,b,c} = q^a (\prod_{i=1}^m (\gamma_i q^b)^{p_i}) q^c$.

PROPOSITION 4.9. *The length of each example in POS(p, k, I) and NEG(p, k, I), and the number of examples in these sets, is polynomial in $|I|$ and in the value p.*

PROOF. The length of q is at most $p^{k+1} n^{k+1}$, since there are at most $p^k n^k$ choices of $\vec{x}$ in the product defining q, and for each choice, the factor in the product is exactly pn characters long. Thus, the single element q^{p^k} of POS(p, k, I) has length $p^k |q| \leq p^{2k+1} n^{k+1}$. Each element of NEG$(p, k, I)$ consists of

—at most $(pm + 2)p^k$ copies of q. (The leading a copies of q, the trailing c copies of q, and pm instances of b copies of q, included with the subwords γ_i.);

—at most pm instances of subwords $\gamma_i = \alpha_i \beta_i$, of q, having total length at most $2pm|q|$.

Thus, the total length of any element of NEG(p,k,I) is at most $((pm+2)p^k + 2pm)|q| \le ((pm+2)p^k + 2pm)p^{k+1}n^{k+1}$. Recalling that $m = 3 \cdot 2^{k-1}$, a constant, each element of NEG(p,k,I) is clearly of length polynomial in $|I|$ and p, and the lemma follows if the number of elements of NEG(p,k,I) is also polynomial in $|I|$ and p. To see that this is the case, observe that the number of elements of NEG(p,k,I) is at most the number of ways to choose m pairs (α_i, β_i), some element of P^m, and three values a, b, and c. The total number of possible examples is thus at most $((|q|+1)^2)^m p^m (p^k)^3$, a polynomial in $|I|$ and p. □

PROPOSITION 4.10. *For any k and p, $R_{p,k}(I)$ is computable in time polynomial in $|I|$ and in the value p.*

PROOF. As shown in the proof of Proposition 4.9, the number of ways to choose m pairs (α_i, β_i), some element of P^m, and three values a, b, and c, is polynomial in $|I|$ and p. Also the number of ways to choose a set D of at most $k-1+m$ variables is polynomial in |I|, since $k-1+m$ is constant. Thus we only need to show that for a particular choice of $\{\alpha_i\}_{i=1}^m$, $\{\beta_i\}_{i=1}^m$, $\langle p_1, p_2, \dots, p_m \rangle$, and D, it is possible to check in time polynomial in $|I|$ and p if $COL(\vec{\gamma}) \subseteq D$ and $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$, where $\gamma = \prod_{i=1}^m (\alpha_i \beta_i)^{p_i}$.

Observe that $\vec{\gamma}$ and $SOL(I|_D)$ can be easily constructed and $COL(\vec{\gamma}) \subseteq D$ is easy to check. Let $\vec{\gamma} = \langle x_1, x_2, \dots x_n \rangle$. We are left with having to determine whether $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$. Since $COL(\vec{\gamma}) \subseteq D$ this is equivalent to determining whether there exists an assignment τ: $D \to \{0,1\}$ of the variables of D, such that either τ violates some clause of $SOL(I|_D)$ or the sum of all components x_i of $\vec{\gamma}$ such that $v_i \in D$ and $\tau(v_i) = 1$ is not equal to zero (mod p). Since both D and $SOL(I|_D)$ have constant size, this method for determining whether $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$ requires only constant time (on a unit cost RAM). □

4.3. A SMALL CONSISTENT COUNTER MACHINE. We show that if I has some solution τ, then there is a small (p state) DFA consistent with POS(p,k,I) and NEG(p,k,I).

PROOF OF LEMMA 4.2. Observe that $C(p,\tau)$ accepts a string γ iff $\tau \in K(\vec{\gamma})$, that is, iff the number of true variables occurring in γ is zero mod p. Since $\vec{q} = \vec{0}$, then for any power q^a of q, $\vec{q}^a = \vec{0}$. Clearly, $\tau \in K(\vec{q}^a)$, and thus $C(p,\tau)$ accepts q^a. In particular, $C(p,\tau)$ accepts the only element q^{p^k} of POS(p,k,I).

To see that $C(p,\tau)$ rejects all elements of NEG(p,k,I), we show that if $C(p,\tau)$ accepts a string $\gamma_{a,b,c}$ (as described in the definition of NEG(p,k,I)), then $\gamma_{a,b,c}$ is not an element of NEG(p,k,I). Note that since q contains each variable occurring a number of times congruent to 0 mod p, if $C(p,\tau)$ accepts $\gamma_{a,b,c}$, then it also accepts any word formed from $\gamma_{a,b,c}$ by removing any number of copies of q. In particular, it also accepts γ, (described in the definition of NEG(p,k,I)). From our comments preceding Section 4.1, it follows that $\tau \in K(\vec{\gamma})$. Further, by assumption $\tau \in SOL(I)$, hence for any

$D \subseteq V$, $\tau \in SOL(I|_D)$. Thus, certainly there is no set $D \subseteq V$ of size $|D| \le k - 1 + m$, such that $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$, and therefore, for no choices of a, b, and c would the string $\gamma_{a,b,c}$ be placed in NEG(p, k, I). □

To prove that p states are necessary in any NFA consistent with POS(p, k, I) and NEG(p, k, I) (Lemma 4.3), we first introduce some notation that will also be used in the proof of Lemma 4.4.

Let $A = (Q, \Sigma, \delta, s_{\text{init}}, F)$ be any NFA with less than p^k states that is consistent with POS(p, k, I) and NEG(p, k, I), and again consider the graph representation of A. Since A is consistent, the positive example q^{p^k} defines a path ψ from s_{init} to some accepting state s_+. Since A has less than p^k states, there exist numbers d, e, and f such that $d + 1 + e + f = p^k$, and states s and t on path ψ such that: q^d leads from s_{init} to s; q leads from s to t; q^e leads from t to s; and q^f leads from s to s_+. Figure 2 shows the loop of ψ, together with states s and t, and the strings q^d, q^e, q^f. (The states s_{init}, s, t, and s_+ need not be distinct.)

For any prefix w of q^{p^k}, let $s_{\text{init}}(w)$ denote the state that w leads to along ψ. Since we assumed (see Section 2) that NFAs do not have λ-transitions, the state $s_{\text{init}}(w)$ is uniquely defined. For each $\vec{x} \in P^n$ for which $|COL(\vec{x})| \le k$ we define a particular prefix $\alpha_{\vec{x}}$ and suffix $\beta_{\vec{x}}$ of q as follows:

$$\alpha_{\vec{x}} = \left(\prod_{\substack{\vec{y} < \vec{x} \\ |COL(\vec{y})| \le k}} v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n} v_1^{p-y_1} v_2^{p-y_2} \cdots v_n^{p-y_n} \right) v_1^{x_1} v_2^{x_2} \cdots v_n^{x_n}$$

and $\beta_{\vec{x}}$ is the unique suffix such that $\alpha_{\vec{x}} \beta_{\vec{x}} = q$. In other words,

$$\beta_{\vec{x}} = v_1^{p-x_1} v_2^{p-x_2} \cdots v_n^{p-x_n} \left(\prod_{\substack{\vec{y} > \vec{x} \\ |COL(\vec{y})| \le k}} v_1^{y_1} v_2^{y_2} \cdots v_n^{y_n} v_1^{p-y_1} v_2^{p-y_2} \cdots v_n^{p-y_n} \right).$$

Proof of Lemma 4.3. We show that any consistent NFA A with less than p^k states must have at least p states. Let n be the number of variables of I and let $\{v_i, v_j, v_k\}$ be any clause of I. Let $\vec{x}_1$ be the n-component vector that contains all 0's except for components i, j, and k, which are 1. Let $\vec{x}_r = r\vec{x}_1$, for $0 \le r \le p - 1$.

It suffices to show that the states $\{s_{\text{init}}(q^d \alpha_{\vec{x}_r})\}_{0 \le r \le p-1}$ are distinct. Assume to the contrary, that for some r and r' such that $0 \le r' < r \le p - 1$, we have $s_{\text{init}}(q^d \alpha_{\vec{x}_r}) = s_{\text{init}}(q^d \alpha_{\vec{x}_{r'}})$. Observe that $q^d \alpha_{\vec{x}_r} \beta_{\vec{x}_r} q^{e+f} = q^d \alpha_{\vec{x}_{r'}} \beta_{\vec{x}_{r'}} q^{e+f} = q^{p^k}$. Since $s_{\text{init}}(q^d \alpha_{\vec{x}_r}) = s_{\text{init}}(q^d \alpha_{\vec{x}_{r'}})$, we conclude that the word $\mu = q^d \alpha_{\vec{x}_r} \beta_{\vec{x}_{r'}} q^e q^f$ is accepted. We now obtain a contradiction by showing that $\mu \in$ NEG(p, k, I). Thus, $s_{\text{init}}(q^d \alpha_{\vec{x}_r}) \ne s_{\text{init}}(q^d \alpha_{\vec{x}_{r'}})$, and A must have at least p states.

To see that μ is a negative example, recall the definition of NEG(p, k, I). Set γ_1 to $\alpha_{\vec{x}_r} \beta_{\vec{x}_{r'}}$ and γ_l to $\alpha_{\vec{x}_r} \beta_{\vec{x}_r} = q$, for $2 \le l \le m$. Let $\langle p_1, p_2, \ldots, p_m \rangle$ be the vector $\langle 1, 0, \ldots, 0 \rangle$ of P^m and set D to be the clause $\{v_i, v_j, v_k\}$. The word μ can be rewritten as $q^d(\prod_{l=1}^m (\gamma_l q^e)^{p_l}) q^f = q^d \gamma_1 q^e q^f$. To show that μ is a negative example we only need to show that $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$ for $\gamma = \prod_{l=1}^m (\gamma_l)^{p_l} = \gamma_1$. Note $\vec{\gamma}$ consists of all zeros except for the components i, j, and k, which have value $r - r'$. The restriction $I|_D$ consists of the clause D and thus any solution of $SOL(I|_D)$ must have exactly one of the three

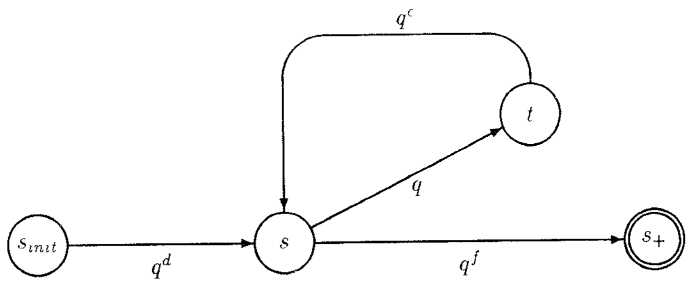

FIG. 2. The loop of path ψ in A.

components corresponding to the variable of the clause set to 1 and the others set to 0. Since $1 \le r - r' \le p - 1$, this implies that such a solution cannot lie in $K(\vec{\gamma})$ and $SOL(I|_D) \cap K(\vec{\gamma}) = \varnothing$. □

4.4. FORCING POLYNOMIALLY LARGER NFAS. In this section, we prove Lemma 4.4 by showing that if there exists an NFA consistent with POS(p, k, I) and NEG(p, k, I) with fewer than p^k states, then there exists a solution of instance I. Before proving Lemma 4.4, we need a number of supporting propositions. The arguments to follow apply to any NFA A that satisfies the hypothesis of Lemma 4.4. As described above, A must have a loop; in what follows we use the notation given in Figure 2.

For any set $X \subseteq \{1, 2, \ldots, n\}$ such that $|X| = k$, we define an X-bridge as follows: The string $\alpha_{\vec{x}}\beta_{\vec{y}}$ is an X-bridge iff $\vec{x} \ne \vec{y}$, $COL(\vec{x})$ and $COL(\vec{y})$ are both subsets of X, and the string $\alpha_{\vec{x}}\beta_{\vec{y}}$ leads from s to t. Figure 3 depicts a bridge.

PROPOSITION 4.11. *For all $X \subseteq \{1, 2, \ldots, n\}$ such that $|X| = k$, there exists an X-bridge.*

PROOF. Consider any X as in the hypothesis of the proposition. There are exactly p^k vectors $\vec{x}$ such that $COL(\vec{x}) \subseteq X$. Since $|Q| < p^k$, there must be two such vectors $\vec{x} \ne \vec{y}$ such that $COL(\vec{x}) \subseteq X$, $COL(\vec{y}) \subseteq X$ and for some state r, the states $s_{\text{init}}(q^k\alpha_{\vec{x}}) = s_{\text{init}}(q^d\alpha_{\vec{y}}) = r$. The string $\alpha_{\vec{y}}\beta_{\vec{y}} = q$ leads from $s = s_{\text{init}}(q^d)$ to $t = s_{\text{init}}(q^{d+1})$, and thus $\beta_{\vec{y}}$ leads from r to t. Consequently, $\alpha_{\vec{x}}\beta_{\vec{y}}$ leads from s to t, and is therefore an X-bridge. □

A *bridge* is a string that for some X of size k, is an X-bridge. By the definition of a bridge, and e, the string $\alpha_{\vec{x}}\beta_{\vec{y}}q^e$ leads from s to t and then back to s. Thus, q^d (which leads from s_{init} to s), followed by any sequence of strings of the form $\alpha_{\vec{x}}\beta_{\vec{y}}q^e$ where $\alpha_{\vec{x}}\beta_{\vec{y}}$ is a bridge, followed by q^f (which leads from s to s_+) is accepted by A (refer to Figure 3). We have just proved the following proposition.

PROPOSITION 4.12. *Let $\{\alpha_i\}_{i=1}^m$ be any collection of prefixes of q, and $\{\beta_i\}_{i=1}^m$ any collection of suffixes of q, such that for each i, $1 \le i \le m$, the string $\gamma_i = \alpha_i\beta_i$ is a bridge. Then for any $\langle p_1, p_2, \ldots, p_m \rangle \in P^m$, and for any string $\gamma = \prod_{i=1}^m(\gamma_i q^e)^{p_i}$, the NFA A accepts the string $q^d\gamma q^f$.*

Recall from the discussion at the beginning of Section 4 that in any counter machine $C(p, \tau)$, if γ is a word that leads from a state back to the same state,

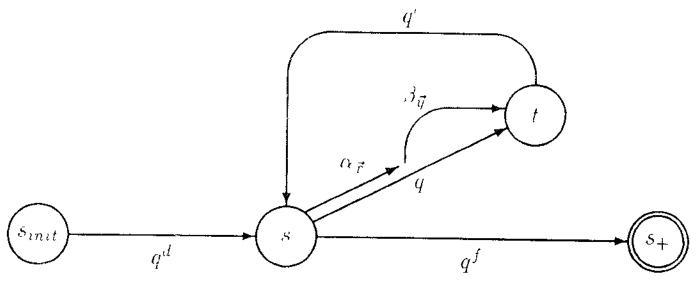

FIG. 3. A bridge from s to t.

then τ satisfies $\vec{\gamma} \cdot \tau = 0$. Since for every bridge $\alpha_{\vec{x}}\beta_{\vec{y}}$ the string $\gamma_{\vec{x}\vec{y}} = \alpha_{\vec{x}}\beta_{\vec{y}}q^e$ forms a cycle in A, we follow the approach discussed earlier, and from $\gamma_{\vec{x}\vec{y}}$ we derive the equation $\vec{\gamma}_{\vec{x}\vec{y}} \cdot \tau = 0$ for the unknown assignment vector τ. Since $\vec{q}^e = \vec{0}$, this equation is equivalent to $(\vec{\alpha}_{\vec{x}} + \vec{\beta}_{\vec{y}}) \cdot \tau = \vec{0}$. Our goal is to extract from A a small collection of bridges S such that at least one solution of the collection of associated equations is also a solution to the instance I. The examples NEG(p, k, I) will rule out collections of bridges whose corresponding sets of equations do not have this property.

Let R be the matrix with a row $\vec{\alpha}_{\vec{x}} + \vec{\beta}_{\vec{y}}$ for each bridge $\alpha_{\vec{x}}\beta_{\vec{y}}$ of A. By Proposition 4.11, for every X of size k there is an X-bridge $\alpha_{\vec{x}}\beta_{\vec{y}}$ such that $COL(\vec{\alpha}_{\vec{x}} + \vec{\beta}_{\vec{y}}) \subseteq X$ and $\vec{x} \neq \vec{y}$ (i.e., row $\vec{\alpha}_{\vec{x}} + \vec{\beta}_{\vec{y}} \neq \vec{0}$). Thus, R is k-closed (Definition 4.5), and therefore, by Lemma 4.6, R contains a core C of size at most $k - 1$. Also, from Lemma 4.6, there exists a determining row for each $i \in \{1, 2, \ldots, n\} - C$.

Definition 4.13. Let S be any matrix of $n - |C|$ rows of R containing exactly one determining row for each index of $\{1, 2, \ldots, n\} - C$.

For any set M of rows of length n, let $V(M)$ abbreviate $V_{COL(M)}$ (the variables v_i for which there is a row in M with a nonzero ith component). Let V_C denote the set of variables indexed by the core, that is, $\{v_i : i \in C\}$. Once the variables of V_C are set, the values of all other variables are forced in the following sense: Given a setting of values for V_C, that is, $\sigma: V_C \to \{0, \ldots, p-1\}$, the determining rows in S for all indices outside of the core ensure that there is exactly one extension of σ to a setting of all variables $\sigma': V \to \{0, \ldots, p-1\}$ that lies in $K(S)$. Since $|C| \leq k - 1$, this implies that there are at most 2^{k-1} elements of $K(S) \cap \{0,1\}^n$. Lemma 4.4 will be proved using the following proposition:

PROPOSITION 4.14. *Consider the numbered statements below*:

(*i*) $SOL(I) \cap K(S) = \varnothing$.

(*ii*) *There exists a subset of rows T of S such that $|T| \leq m = 3 \cdot 2^{k-1}$, and such that $SOL(I|_{V_C \cup V(T)}) \cap K(T) = \varnothing$.*

(*iii*) *There exists an element $\vec{\omega}$ of span(T) such that $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega}) = \varnothing$.*

Then (*i*) $\Rightarrow$ (*ii*) $\Rightarrow$ (*iii*).

PROOF THAT (i) IMPLIES (ii). This portion of the proposition asserts that if the system of equations S has no solution in common with the set of solutions of I, then a constant-sized subset T of S has no solutions in common with the set of solutions of I *restricted* (see Definition 4.8) to those variables V_C of the core, and those variables $V(T)$ with nonzero entries in some row of T.

We construct a matrix T from a subset of the rows of S such that T has the desired properties. For each assignment τ: $V_C \to \{0,1\}$, there is exactly one extension of τ to a setting of all variables τ': $V \to \{0, \dots, p-1\}$ that lies in $K(S)$. Since $SOL(I) \cap K(S) = \emptyset$, we have that either

(1) there is a number $i \in \{1, 2, \dots, n\} - C$ such that $\tau'(v_i) \notin \{0,1\}$, or
(2) $\tau' \in \{0,1\}^n \cap K(S)$ but $\tau' \notin SOL(I)$.

In case 1, include a determining row $\vec{x}$ of index i in T (such a row exists in S).

In case 2, the unique extension τ' of τ is an element of $\{0,1\}^n \cap K(S)$ but there exists a clause $(v_j, v_{j'}, v_{j''})$ of I such that the multiset $\{\tau'(v_j), \tau'(v_{j'}), \tau'(v_{j''})\} \neq \{0,0,1\}$. That is, it is not the case that exactly one variable is set to true. Then, for each $i \in \{j, j', j''\} - C$, include in T the determining row $\vec{x}$ of i in S.

Clearly, $|T| \leq m = 3 \cdot 2^{k-1}$, since for each assignment τ of V_C (and $|C| \leq k-1$), at most three rows of S were included in T.

We show that $SOL(I|_{V_C \cup V(T)}) \cap K(T) = \emptyset$. Suppose this is not the case, and there exists a vector $\vec{z} \in SOL(I|_{V_C \cup V(T)}) \cap K(T)$. Let τ be the assignment of the variables of V_C given by $\vec{z}$, that is, for all $i \in C$, $\tau(v_i) = z_i$. Then τ was considered in the above construction of T. Since the unique extension τ' of τ to V that lies in $K(S)$ cannot be an element of $SOL(I)$, one of the two cases occurred.

If case 1 occurred, then there is a determining row $\vec{x}$ of T and a number i such that any extension τ'' of τ to an assignment of V for which $\vec{x} \cdot \tau'' = 0$ must be such that $\tau''(v_i)$ is a unique value that is not in $\{0,1\}$. Since $\vec{x} \in T \subseteq S$, it follows that both τ' and $\vec{z}$ are such extensions and thus $\tau'(v_i) = z_i \notin \{0,1\}$. But the fact that $z_i \notin \{0,1\}$ contradicts the fact that $\vec{z} \in SOL(I|_{V_C \cup V(T)}) \cap K(T)$.

If case 2 held, then for the unique extension of τ to $\tau' \in \{0,1\}^n \cap K(S)$, there exists a clause $(v_j, v_{j'}, v_{j''})$ of I such that the multiset $\{\tau'(v_j), \tau'(v_{j'}), \tau'(v_{j''})\} \neq \{0,0,1\}$. If $i \in \{j, j', j''\} - C$, then the determining row $\vec{x}$ of i was included in T, and any extension of τ to some $\tau'' \in \{0,1\}^n \cap K(T)$ must set the variable v_i to the same value as τ' does. On the other hand, if $i \in \{j, j', j''\} \cap C$, then since τ has domain V_C, any extension τ'' of τ to an assignment of V must set the variable v_i to the same value as τ' does. Thus, regardless of whether $i \in \{j, j', j''\} \cap C$ or $i \in \{j, j', j''\} - C$, we have constructed T so that $\{v_j, v_{j'}, v_{j''}\} \subseteq V_C \cup V(T)$, and such that for each $v_i \in \{v_j, v_{j'}, v_{j''}\}$, any extension of τ to $\tau'' \in \{0,1\}^n \cap K(T)$ is such that $\tau''(v_i) = \tau'(v_i)$. Since $\vec{z}$ is such an extension of τ, it follows that the multiset $\{z_j, z_{j'}, z_{j''}\} = \{\tau'(v_j), \tau'(v_{j'}), \tau'(v_{j''})\} \neq \{0,0,1\}$ and thus $\vec{z} \notin SOL(I|_{V_C \cup V(T)})$, and certainly $\vec{z} \notin SOL(I|_{V_C \cup V(T)}) \cap K(T)$, as was hypothesized above. This completes the proof that (i) implies (ii).

PROOF THAT (ii) IMPLIES (iii). Suppose to the contrary that (ii) holds but that (iii) does not, so no such element $\vec{\omega}$ exists. Then for each element $\vec{\omega}$ of

$span(T)$, $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega})$ is nonempty. Since $V(\vec{\omega}) \subseteq V(T)$, if τ: $V_C \cup V(T) \to \{0, 1\}$, then either all extensions τ': $V \to \{0, 1\}$ of τ are such that $\tau' \in SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega})$, or no extensions τ' are such that $\tau' \in SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega})$. In the first case, we say that τ is a *witness* to the nonemptiness of $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega})$. Since for each $\vec{\omega}$, we have $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega}) \neq \emptyset$, there is a witness for each $\vec{\omega}$. There are at most 2^{k-1+m} distinct witnesses, since each is an assignment of at most $k - 1 + m$ variables (at most $k - 1$ variables of the core C and at most one variable not in C for each of the m determining rows of S that are in T). Thus, there must be some assignment τ_0: $V_C \cup V(T) \to \{0, 1\}$ that is a witness for at least

$$\frac{|span(T)|}{2^{k-1+m}} > \frac{|span(T)|}{p}$$

elements of $span(T)$ (because $p > 2^{k-1+m}$). Let B be the subset of elements of $span(T)$ for which τ_0 is a witness, that is, B is the set of elements of $span(T)$ such that for every $\vec{\omega} \in B$, every extension of τ_0 is an element of $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega})$. Immediately, we have that every extension of τ_0 is an element of $SOL(I|_{V_C \cup V(T)}) \cap K(B)$.

By Lemma 4.7, the above bound on the size of B implies that B contains a basis of T and thus $K(T) = K(B)$. Thus, $SOL(I|_{V_C \cup V(T)}) \cap K(T) = SOL(I|_{V_C \cup V(T)}) \cap K(B)$. But since every extension of τ_0 is in $SOL(I|_{V_C \cup V(T)}) \cap K(B)$, this implies that $SOL(I|_{V_C \cup V(T)}) \cap K(T)$ is nonempty, contradicting the hypothesis of the proposition. □

PROOF OF LEMMA 4.4. Assume the hypothesis of the lemma is true; we need only show that there exists a solution of instance I. Thus, Lemma 4.4 follows immediately from:

PROPOSITION 4.15. $SOL(I) \cap K(S) \neq \emptyset$.

PROOF. Suppose to the contrary that $SOL(I) \cap K(S) = \emptyset$. Then, by Proposition 4.14, there exists $T \subseteq S$ of size at most m such that $SOL(I|_{V_C \cup V(T)}) \cap K(T) = \emptyset$. This implies, again by Proposition 4.14, that there exists a vector $\vec{\omega} \in span(T)$ such that $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\omega}) = \emptyset$.

Recall that T consists of $l \leq m$ vectors $\vec{\gamma}_1, \vec{\gamma}_2, \ldots, \vec{\gamma}_l$, corresponding to bridges $\gamma_1, \gamma_2, \ldots, \gamma_l$, where each γ_i is formed from the concatenation of some prefix α_i and some suffix β_i of q. For syntactic convenience, define $\gamma_j = \gamma_1$ for $l < j \leq m$. By the definition of $span(T)$, there exists $\vec{p} \in P^m$ such that $\vec{\omega} = \Sigma_{i=1}^{m} p_i \vec{\gamma}_i$. Let the string $\gamma = \Pi_{i=1}^{m} (\gamma_i)^{p_i}$. Then, clearly $\vec{\gamma} = \vec{\omega}$, and thus $SOL(I|_{V_C \cup V(T)}) \cap K(\vec{\gamma}) = \emptyset$. Now define $D = V_C \cup V(T)$, and we have that $SOL(I|_D) \cap K(\vec{\gamma}) = \emptyset$, $|D| \leq k - 1 + m$, and $COL(\vec{\gamma}) \subseteq D$. Then, by the definition of NEG(p, k, I), for any a, b, c, $\gamma_{a,b,c} = q^a(\Pi_{i=1}^{m}(\gamma_i q^b)^{p_i})q^c$ is an element of NEG(p, k, I). But when $a = d$, $b = e$, and $c = f$, by Proposition 4.12, this string is accepted by A, contradicting the consistency of A. We conclude that $SOL(I) \cap K(S) \neq \emptyset$, completing the proof of Proposition 4.15 and Lemma 4.4. □

5. *A Larger than Polynomial Gap*

We extend Theorem 4.1 to show that not only is a polynomial approximation factor unachievable, but no polynomial-time approximation algorithm can find a consistent NFA of size $opt^{(1-\epsilon)\log\log opt}$ for any constant $\epsilon > 0$. (In this paper, "log" denotes "$\log_2$.")

THEOREM 5.1. *For any constant* $\epsilon > 0$, *MIN-CON(DFA, NFA) is not* $opt^{(1-\epsilon)\log\log opt}$*-approximable unless P = NP.*

PROOF. Suppose to the contrary that there exists a polynomial-time algorithm A, a constant $\epsilon > 0$, and a constant c such that on input of any instance of MIN-CON(DFA, NFA) for which the smallest consistent DFA has at least $opt \geq c$ states, A outputs a consistent NFA with at most $opt^{(1-\epsilon)\log\log opt}$ states.

We obtain a contradiction by applying the reduction $R_{p,k}$ to an instance I of a 1-in-3-SAT problem for appropriate choice of p and k. Choose any prime $p > \max\{c, 2^{2^{3/\epsilon}}\}$. We show below that for $k = \lfloor \log\log p \rfloor - 2$, the following two conditions hold:

(1) $p^{(1-\epsilon)\log\log p} < p^k$,
(2) $p > 2^{k-1+m}$.

If we apply $R_{p,k}$ to instance I, we obtain sets POS(p, k, I) and NEG(p, k, I) for which the number of states in a smallest DFA is $p > c$ if I has a solution, or is at least p^k if I does not (Lemmas 4.2, 4.3, and 4.4).

Certainly if I has no solution, then approximation algorithm A cannot find an NFA with fewer than p^k states. On the other hand, if I has a solution, then A must find an NFA with at most $p^{(1-\epsilon)\log\log p} < p^k$ states. By the second condition above, p satisfies the appropriate size bound and the proof of Theorem 4.1 applies, so A may be used to solve 1-in-3-SAT in polynomial time.

To complete the proof of Theorem 5.1 we must show that the two conditions above hold. To see that the first condition holds, note that by our choice of p, $\epsilon \log\log p > 3$, and so $(1-\epsilon)\log\log p < \log\log p - 3 < \lfloor \log\log p \rfloor - 2$. Thus, $p^{(1-\epsilon)\log\log p} < p^{\lfloor \log\log p \rfloor - 2} = p^k$.

For the second condition, assume without loss of generality that $\epsilon < 1$, thus $p > 2^8$ and

$$\begin{aligned}
p &> (\log p)p^{1/2} \\
&= 2^{\log\log p + 2^{\log\log p - 1}} \\
&= 2^{\log\log p + 4\cdot 2^{\log\log p - 3}} \\
&> 2^{\lfloor \log\log p \rfloor - 3 + 3\cdot 2^{\lfloor \log\log p \rfloor - 3}} \\
&= 2^{k-1+3\cdot 2^{k-1}} \\
&= 2^{k-1+m},
\end{aligned}$$

completing the proof of Theorem 5.1. □

Let REGEXPR and REGGRAM denote the classes of regular expressions and regular grammars, respectively.

COROLLARY 5.2. *For any constant* $\epsilon > 0$, *MIN-CON(DFA, REGEXPR) and MIN-CON(DFA, REGGRAM) are not* $opt^{(1-\epsilon)\log\log opt}$*-approximable unless P = NP.*

PROOF. By trivially modifying the proof of Theorem 5.1 and using Lemmas 2.3 and 2.5, it follows that MIN-CON(DFA, REGEXPR) and MIN-CON (DFA, REGGRAM) are not $\frac{1}{2}opt^{(1-\epsilon')\log\log opt}$-approximable for any $\epsilon' > 0$

unless P = NP. To see that they are not $opt^{(1-\epsilon)\log\log opt}$-approximable for any $\epsilon > 0$, choose any $\epsilon' < \epsilon$ (and greater than 0), and then for sufficiently large values of opt we have $opt^{(1-\epsilon)\log\log opt} < \frac{1}{2}opt^{(1-\epsilon')\log\log opt}$. Thus, an algorithm guaranteeing an approximation of $opt^{(1-\epsilon)\log\log opt}$ would immediately guarantee an approximation of at most $\frac{1}{2}opt^{(1-\epsilon')\log\log opt}$, which we have observed is not possible for any $\epsilon' > 0$. □

It is important to point out that Theorem 5.1 and Corollary 5.2 hold only when the size of a DFA or NFA is the number of states of the automaton. These results do not hold if we take the size measure to be the number of bits to encode the automaton—an arguably more natural measure. (It is easily shown that Theorem 4.1 still holds for this more natural size measure.)

To see where the theorem fails to apply, first note that the reduction $R_{p,k}$ takes time at least $\Omega(p^k\binom{n}{k})$ when applied to an instance of 1-in-3-SAT of n variables, because the word q^{p^k} alone is at least that long. Thus, in order to use $R_{p,k}$ to solve an instance I of 1-in-3-SAT, k must not be chosen to be any increasing function of n, otherwise the time taken by the reduction would be more than polynomial in n.

A DFA of p states with an n symbol alphabet requires roughly $pn \log pn$ bits to encode, so to achieve a result comparable to Theorem 5.1 for this size measure, it would be necessary to show the difficulty of finding an NFA with a description of at most $(pn \log pn)^{(1-\epsilon)\log\log(pn\log pn)}$ bits. To do this would simultaneously show the difficulty of finding an NFA with fewer than, say, $(pn)^{\log\log\log pn}$ states, so the value k in the reduction would need to be at least $\log\log pn$, an increasing function of n, and hence the reduction would not be polynomial in the size of the instance I.

In Sections 6, 7, and 8, we consider some related MIN-CON problems and prove approximability lower bounds of opt^k for any constant k, as in Theorem 4.1. However, we will not be able to extend the lower bounds to $opt^{(1-\epsilon)\log\log opt}$ for the following reasons. The proofs of the theorems in the following sections will necessitate applications of reductions $R_{p,k}$ to an instance I of n variables, where p is chosen to be at least n. If we try to apply the proof of Theorem 5.1, we note that the value k is chosen to be roughly $\log\log p$. Consequently, the exponent k of the reduction would grow as a function of n, and the reduction would not be executable in time polynomial in n.

6. *The Two-Letter Case*

We generalize Theorem 4.1 to the case of DFAs and NFAs over the two-letter alphabet $\{0,1\}$. Let $\mathrm{DFA}^{\{0,1\}}$ and $\mathrm{NFA}^{\{0,1\}}$ denote the class of DFAs and NFAs over alphabet $\{0,1\}$, respectively.

THEOREM 6.1. *If $P \neq NP$, then for all positive integers k, MIN-CON $(DFA^{\{0,1\}}, NFA^{\{0,1\}})$ is not opt^k-approximable.*

COROLLARY 6.2. *If $P \neq NP$, then MIN-CON$(DFA^{\{0,1\}}, NFA^{\{0,1\}})$ is not $f(opt)$-approximable for any function f that is bounded above by some polynomial.*

The proof of Theorem 6.1 (presented later) is essentially the same as that of Theorem 4.1, except that, since all examples must be constructed using only two symbols, binary sequences are used to encode the symbols of the alphabet.

Let I be an instance of 1-in-3-SAT with variable set $V = \{v_1, v_2, \dots, v_n\}$. Without loss of generality, n is a power of 2, for otherwise we could pad by adding dummy variables. Fix some bijection "*hat*" (we write "$\hat{v}$" to denote "*hat*(v)") with domain V and range $\{0,1\}^{\log n}$, and thus $\hat{v}_i$ is the bit string of length $\log n$ that "encodes" the variable v_i. Inductively extend *hat* in the usual way to strings of $V^+ = V^* - \{\lambda\}$ by letting $\widehat{wv} = \hat{w}\hat{v}$ where $w \in V^+$ and $v \in V$.

Let $R_{p,k}^{\{0,1\}}$ be the (polynomial-time) algorithm that takes as input an instance I of 1-in-3-SAT, and outputs the sets $\widehat{\text{POS}}(p,k,I) = \{\hat{w} : w \in \text{POS}(p,k,I)\}$ and $\widehat{\text{NEG}}(p,k,I) = \{\hat{w} : w \in \text{NEG}(p,k,I)\}$. (That there is such a polynomial-time $R_{p,k}^{\{0,1\}}$ follows immediately from the definition of the function *hat*, and from Proposition 4.10.)

If $\tau \colon V \to \{0,1\}$ is any assignment, then define $\hat{C}(p,\tau)$ to be the "counter-like" machine over the alphabet $\{0,1\}$ that, on input $\hat{w}$, simulates $C(p,\tau)$ on input $w : \hat{C}(p,\tau)$ will accept the language $\hat{L}(C(p,\tau)) = \{\hat{w} : w \in L(C(p,\tau))\}$ and may be constructed from $C(p,\tau)$ as follows: Let δ be the transition function of $C(p,\tau)$. Then, $\hat{C}(p,\tau)$ consists of *counter* states and *auxiliary states*, and has transition function $\hat{\delta}$. The counter states are exactly the states of $C(p,\tau)$. We replace each of the n edges emanating from any counter state s with a complete binary tree of depth $\log n$. The tree is rooted at s, and each vertex has edges labeled 0 and 1 to its left and right child, respectively. All $(n-2)$ internal vertices other than the root are auxiliary (new) states, and the leaves are counter states such that for each $v \in V$, $\hat{\delta}(s,\hat{v}) = \delta(s,v)$.

LEMMA 6.3. *If I has a solution, then for any positive integers k and p, the size opt of the smallest DFA consistent with $\widehat{POS}(p,k,I)$ and $\widehat{NEG}(p,k,I)$ satisfies $p \le opt \le pn$, where n is the number of variables of instance I.*

PROOF. The first inequality follows easily from the proof of Lemma 4.3, which shows that the smallest DFA consistent with $\text{POS}(p,k,I)$ and $\text{NEG}(p,k,I)$ has at least p states. To prove the second inequality, observe that by the construction of $\hat{C}(p,\tau)$ and by Lemma 4.2, if I has some solution τ, then $\hat{C}(p,\tau)$ is a DFA consistent with $\widehat{\text{POS}}(p,k,I)$ and $\widehat{\text{NEG}}(p,k,I)$ and has exactly $p + p(n-2) \le pn$ states. Note that the lower bound of the lemma holds for NFAs, and the upper bound holds for DFAs.

LEMMA 6.4. *Let k be any positive integer, and let p be a prime such that $p > 2^{k-1+m}$. If I is any instance of 1-in-3-SAT, and if there exists an NFA A with less than p^k states that is consistent with $\widehat{POS}(p,k,I)$ and $\widehat{NEG}(p,k,I)$, then I has some solution.*

PROOF. If the hypothesis is true we may easily obtain an NFA A' over alphabet V, and with at most as many states as A, that is consistent with $\text{POS}(p,k,I)$ and $\text{NEG}(p,k,I)$. Thus, by Lemma 4.4, I has some solution. □

Now to prove Theorem 6.1, define, for each k, a transformation T_k as follows. On input instance I of a 1-in-3-SAT problem with n variables, T_k first determines whether $n > 2^{k-1+m}$. If not, then T_k halts and outputs nothing. Otherwise, T_k finds the smallest prime number p satisfying $2^{k-1+m} < n \le p \le 2n$. Such a p exists by a theorem of Chebyshev (Theorem 8.6, page 185 of [19]), which states that for all $n > 1$ there exists a prime p such that $n < p < 2n$. After obtaining p, T_k then computes and outputs $R_{p,k}^{\{0,1\}}(I)$.

We argue that for each constant k, T_k is computable in time polynomial in $|I|$. Since $|I| \geq n$, checking the $n + 1$ numbers p between n and $2n$ for primality may be done in time polynomial in $|I|$ by trying all possible divisors up to $\sqrt{p}$. (Note that the time is polynomial in n, and not in $\log n$). By Proposition 4.10, the run time of $R_{p,k}^{\{0,1\}}$ is polynomial in $|I|$ as well as the value p, which is at most $2n$.

PROOF OF THEOREM 6.1. Let k be given. Then there exists a number k' such that $opt^k < (opt/2)^{k'/2}$ for all sufficiently large values of opt. Consequently, to show for all k that MIN-CON(DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$) is not opt^k-approximable unless P = NP, it suffices to show for all k that MIN-CON (DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$) is not $(opt/2)^{k/2}$-approximable unless P = NP.

Suppose to the contrary that for some number k, MIN-CON (DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$) was $(opt/2)^{k/2}$-approximable, witnessed by constant c and polynomial-time algorithm APPROX. We show that membership in 1-in-3-SAT is decidable in polynomial time, hence P = NP, thus proving the theorem.

The polynomial-time algorithm DECIDE determines 1-in-3-SAT as follows: On input instance I with n variables, DECIDE determines if $n > \max\{c, 2^{k-1+m}\}$. If not, then DECIDE determines whether I has a solution by trying all possible assignments (a constant number, since c, k, and m are constants). Otherwise, DECIDE computes $T_k(I)$, and gives the result as input to subroutine APPROX. If APPROX returns an NFA with less than n^k states, then DECIDE outputs "I has a solution." Otherwise, DECIDE outputs "I does not have a solution."

Clearly, DECIDE runs in time polynomial in $|I|$, since both T_k and APPROX run in polynomial time. We must show that DECIDE is correct. Certainly DECIDE is correct for all instances I with $n \leq \max\{c, 2^{k-1+m}\}$ variables, since these decisions are made by exhaustive search.

Suppose I is an instance with $n > \max\{c, 2^{k-1+m}\}$ variables, and that DECIDE outputs "I has a solution." Then, $T_k(I)$ gives the MIN-CON (DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$) instance $R_{p,k}^{\{0,1\}}(I)$ as input to APPROX (where p is the smallest prime between n and $2n$). Since APPROX returns an NFA with less than $n^k \leq p^k$ states, by Lemma 6.4, I has a solution.

Conversely, suppose that I is an instance with $n > \max\{c, 2^{k-1+m}\}$ variables, and I has a solution. Since n is large enough, DECIDE runs $T_k(I)$ and obtains the instance $R_{p,k}^{\{0,1\}}(I) = \widetilde{\text{POS}}(p,k,I), \widetilde{\text{NEG}}(p,k,I)$ of MIN-CON (DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$), where p is the smallest prime between n and $2n$.

By Lemma 6.3, the size opt of the smallest DFA consistent with $\widetilde{\text{POS}}(p,k,I)$ and $\widetilde{\text{NEG}}(p,k,I)$ satisfies $p \leq opt \leq pn$. Since $c < n \leq p \leq opt$, the value of opt is large enough so that the bound on the performance of APPROX must apply, thus APPROX must find a consistent NFA with less than $(opt/2)^{k/2}$ states. Since $opt \leq pn$ and $p \leq 2n$, this gives $opt \leq 2n^2$. Consequently, the consistent NFA returned by APPROX has less than $(2n^2/2)^{k/2} = n^k$ states, and DECIDE outputs "I has a solution." □

7. *Other Representations of Regular Sets*

Let REGGRAM$^{\{0,1\}}$ and REGEXPR$^{\{0,1\}}$ denote the sets of regular grammars and regular expressions, respectively, over the two letter alphabet $\{0,1\}$. We show that Theorem 6.1 implies that unless P = NP, MIN-CON

(DFA$^{\{0,1\}}$, REGGRAM$^{\{0,1\}}$) and MIN-CON(DFA$^{\{0,1\}}$, REGEXPR$^{\{0,1\}}$) are not polynomially approximable.

THEOREM 7.1. *If $P \neq NP$, and f is any function that is bounded above by a polynomial, then MIN-CON(DFA$^{\{0,1\}}$, REGGRAM$^{\{0,1\}}$) and MIN-CON(DFA$^{\{0,1\}}$, REGEXPR$^{\{0,1\}}$) are not $f(opt)$-approximable.*

The proof of this theorem is a modification of the proof of Theorem 6.1. We use the same polynomial-time algorithm $R_{p,k}^{\{0,1\}}$, which for a given instance I of 1-in-3-SAT (with variable set $V = \{v_1, \dots, v_n\}$, where n is a power of 2), produces the sets $\widehat{\text{POS}}(p,k,I) = \{\hat{w} : w \in \text{POS}(p,k,I)\}$ and $\widehat{\text{NEG}}(p,k,I) = \{\hat{w} : w \in \text{NEG}(p,k,I)\}$. Recall that $\hat{v}_i$ is a bit string of length $\log n$ that encodes the variable v_i.

LEMMA 7.2. *Let k be any positive integer, and let p be a prime such that $p > 2^{k-1+m}$. If I is any instance of 1-in-3-SAT, and if there is a regular grammar or a regular expression of size less than $p^k/2$ that is consistent with $\widehat{POS}(p,k,I)$ and $\widehat{NEG}(p,k,I)$, then I has some solution.*

PROOF. Assume there is a consistent regular grammar or a consistent regular expression of size less than $p^k/2$. Then by Lemmas 2.5 and 2.3, respectively, there is a consistent NFA of size less than p^k. We conclude by Lemma 6.4 that I has a solution. □

To prove Theorem 7.1, we use the same transformation T_k, which was used in Section 6. Upon input of some instance I of 1-in-3-SAT with $n > 2^{k-1+m}$ variables, T_k computes the smallest prime number p satisfying $n \leq p \leq 2n$ and runs $R_{p,k}^{\{0,1\}}(I)$. Recall that T_k is computable in time polynomial in $|I|$.

PROOF OF THEOREM 7.1. Since f is bounded above by some polynomial, there exists a number k such that $f(opt) < (opt/8)^{k/2}$ for all sufficiently large values of opt. Thus, it is sufficient to prove that for all positive integers k, neither MIN-CON(DFA$^{\{0,1\}}$, REGGRAM$^{\{0,1\}}$) nor MIN-CON(DFA$^{\{0,1\}}$, REGEXPR$^{\{0,1\}}$) are $(opt/8)^{k/2}$-approximable unless P = NP.

Suppose, for some positive integer k, that MIN-CON(DFA$^{\{0,1\}}$, REGGRAM$^{\{0,1\}}$) or MIN-CON(DFA$^{\{0,1\}}$, REGEXPR$^{\{0,1\}}$) was $(opt/8)^{k/2}$-approximable, witnessed by constant c and polynomial-time algorithm APPROX. Then the following algorithm DECIDE determines 1-in-3-SAT in polynomial time, thus proving the theorem by contradiction. On input instance I with n variables, DECIDE determines if $n > \max\{c, 2^{k-1+m}\}$. If not, then DECIDE determines whether I has a solution by trying all possible assignments (a constant number, since c, k, and m are constants). Otherwise, DECIDE computes $T_k(I)$, and gives the result as input to subroutine APPROX. If APPROX returns a regular grammar or a regular expression of size less than $n^k/2$, then DECIDE outputs "I has a solution." Otherwise, DECIDE outputs "I does not have a solution."

Clearly DECIDE runs in time polynomial in $|I|$, since both T_k and APPROX run in polynomial time. We must show that DECIDE is correct. Certainly, DECIDE is correct for all instances I with $n \leq \max\{c, 2^{k-1+m}\}$ variables, since these decisions are made by exhaustive search.

Suppose I is an instance with $n > \max\{c, 2^{k-1+m}\}$ variables, and that DECIDE outputs "I has a solution." Then, $T_k(I)$ gives the output of $R_{p,k}^{\{0,1\}}(I)$

as input to APPROX (where p is the smallest prime between n and $2n$). Since APPROX returns a regular grammar or a regular expression of size less than $n^k/2 \le p^k/2$, it follows by Lemma 7.2 that I has a solution.

Conversely, suppose that I is an instance with $n > \max\{c, 2^{k-1+m}\}$ variables, and I has a solution. Since n is large enough, DECIDE runs $T_k(I)$ and obtains the instance $R_{p,k}^{\{0,1\}}(I) = \widetilde{\mathrm{POS}}(p,k,I), \widetilde{\mathrm{NEG}}(p,k,I)$ of MIN-CON(DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$), where p is the smallest prime between n and $2n$.

By Lemma 6.3, the size *opt* of the smallest DFA consistent with $\widetilde{\mathrm{POS}}(p,k,I)$ and $\widetilde{\mathrm{NEG}}(p,k,I)$ satisfies $p \le opt \le pn$. Since $c < n \le p \le opt$, the value of *opt* is large enough so that the bound on the performance of APPROX must apply; thus, APPROX must find a consistent regular grammar or regular expression of size less than $(opt/8)^{k/2}$. Since $opt \le pn$ and $p \le 2n$, $opt \le 2n^2$. Consequently, the consistent regular grammar or regular expression returned by APPROX has size less than $(2n^2/8)^{k/2} \le n^k/2$ and thus DECIDE outputs "I has a solution." □

Because Theorem 7.1 shows polynomial nonapproximability, the theorem holds also with respect to other natural size measures than the ones introduced in Section 2.1, as long as these other size measures are polynomially related to the ones used here.

By "reversing" the construction of Lemma 2.5, it is easily seen that there exists a consistent regular grammar of size at most a constant times larger than the smallest consistent DFA over alphabet $\{0,1\}$. However, in the case of regular expressions, Theorem 7.1 may be uninteresting for unrestricted DFAs over alphabet $\{0,1\}$ if there are finite sets of positive and negative examples such that the smallest consistent regular expression over alphabet $\{0,1\}$ is exponentially larger than the smallest consistent DFA over the same alphabet.[2] In this case, Theorem 7.1 would trivially hold, even without the complexity theoretic assumption that P ≠ NP. However, it is easily verified that, in the case of counter-like DFAs, which were used in the reduction of the previous section (Lemma 6.3), the smallest consistent regular expression is at most polynomially larger than the counter-like DFA $\hat{C}(p,\tau)$.

LEMMA 7.3. *For any p and τ: $\{v_1,\ldots,v_n\} \to \{0,1\}$, the smallest regular expression for the language $L(\hat{C}(p,\tau))$ has size at most $cpn\log n$, for some constant c.*

PROOF. Assume without loss of generality that $\tau(v_1) = \tau(v_2) = \cdots = \tau(v_s) = 0$ and that $\tau(v_{s+1}) = \cdots = \tau(v_n) = 1$. Let τ_0 denote the regular expression $(\hat{v}_1 + \hat{v}_2 + \cdots + \hat{v}_s)$ if $s > 0$ and λ if $s = 0$. (Note that we have omitted superfluous parentheses in accordance with the standard precedence rules [15].) Let τ_1 denote the regular expression $(\hat{v}_{s+1} + \hat{v}_{s+2} + \cdots + \hat{v}_n)$ if $s < n$ and λ if $s = n$. For any regular expression r, let r^p be an abbreviation for the regular expression $rr\cdots r$ (concatenated p times). Then the regular expression

$$(\tau_0)^*\big(((\tau_0)^*\tau_1)^p(\tau_0)^*\big)^*$$

denotes the language $L(\hat{C}(p,\tau))$ and has size $O(pn\log n)$. □

[2] In [8], it is shown that there are *languages* such that the smallest regular expression is exponentially larger than the smallest DFA for the language. It is not clear whether this implies the same separation between the sizes of DFAs and regular expressions consistent with a *finite set of examples*.

8. *Forcing Large Linear Grammars*

Let LIN be the class of linear grammars, as defined in Section 2. By employing techniques similar to those of the previous sections, we obtain nonapproximability results for MIN-CON(LIN, LIN). For simplicity, we only develop the proof for the case of linear grammars where the alphabet is a parameter to the problem, as opposed to presenting results for the fixed alphabet $\{0, 1\}$.

THEOREM 8.1. *If $P \neq NP$, then MIN-CON(LIN, LIN) is not $f(opt)$-approximable, for any function f that is bounded above by some polynomial.*

To prove Theorem 8.1, we essentially repeat the proof of Theorem 4.1, but in the context of linear grammars. Recall that the size of a linear grammar is the total number of symbols on all left and right sides of all productions. A reduction will be exhibited that produces a gap: the smallest consistent grammar will be of size between $p/3$ and $6pn$, (where n is the number of variables of an input instance of 1-in-3-SAT), and whenever there is a grammar of size less than $(p^k - 4)/6$, then the corresponding instance of 1-in-3-SAT will have a solution. The proof of Theorem 8.1 is delayed for the proof of the necessary supporting lemmas.

Let I be an instance of 1-in-3-SAT over variable set V, and let # be an additional symbol not appearing in V. For each $j, 1 \leq j \leq p^k + 1$, define $\mathrm{POS}^j(p, k, I)\#$ and $\mathrm{NEG}(p, k, I)\#$ as follows:

$$\mathrm{POS}^j(p, k, I)\# = \{(q^{p^k})^j\#\};$$
$$\mathrm{NEG}(p, k, I)\# = \{w\# : w \in \mathrm{NEG}(p, k, I)\}.$$

Also, the counter machine $C(p, \tau)\#$ is defined as the counter machine $C(p, \tau)$, with additional transition $\delta(s, \#) = s$ added for each state s.

For each number j, it is easily verified that the proof of Theorem 4.1 holds with only trivial modification if the reduction $R_{p,k}$ outputs $\mathrm{POS}^j(p, k, I)\#$ and $\mathrm{NEG}(p, k, I)\#$ instead of $\mathrm{POS}(p, k, I)$ and $\mathrm{NEG}(p, k, I)$. More specifically, the following lemma holds, which incorporates modified versions of Lemmas 4.2, 4.3, and 4.4.

LEMMA 8.2. *Let I be any instance of 1-in-3-SAT, let k and p be any positive integers, and let j be such that $1 \leq j \leq p^k + 1$.*

(1) *If τ is a solution of I, then $C(p, \tau)\#$ is consistent with $POS^j(p, k, I)\#$ and $NEG(p, k, I)\#$.*
(2) *If A is an NFA that is consistent with $POS^j(p, k, I)\#$ and $NEG(p, k, I)\#$, then A has at least p states.*
(3) *If $p > 2^{k-1+m}$ is prime, and if A is an NFA with less than p^k states that is consistent with $POS^j(p, k, I)\#$ and $NEG(p, k, I)\#$, then I has some solution.*

We define the polynomial-time transformation $R^{\mathrm{lin}}_{p,k}$, which takes as input an instance I of 1-in-3-SAT, and outputs two finite sets $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$ with the required gap properties described in Lemmas 8.3, 8.4, and 8.5 (given below). We present $R^{\mathrm{lin}}_{p,k}(I)$ by describing the sets $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$. It is easily verified that $R^{\mathrm{lin}}_{p,k}$ runs in time polynomial in $|I|$ and the value p. The sets $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$ are constructed so that from any linear grammar of size less than $(p^k - 4)/6$, we can obtain a *right linear* grammar that is consistent with $\mathrm{POS}^j(p,k,I)\#$ and $\mathrm{NEG}(p,k,I)\#$ of size less than $p^k/2$, for some j between 1 and p^k+1. Applying Lemma 2.5, we obtain an NFA with less than p^k states that is consistent with $\mathrm{POS}^j(p,k,I)\#$ and $\mathrm{NEG}(p,k,I)\#$; thus, we obtain, by the third part of Lemma 8.2, a solution of I.

Let $u = (q^{p^k})^{p^k+1}$ and $v = \$^{p^k}¢^{p^k}$, where \$ and ¢ are not elements of V. Then define

$\mathrm{POS}^{\mathrm{lin}}(p,k,I) = \{uv, v\}$,
$\mathrm{NEG}^{\mathrm{lin}}(p,k,I) = \mathrm{M1} \cup \mathrm{M2} \cup \mathrm{M3} \cup \mathrm{M4}$, each defined below.

$\mathrm{M1} = \{xw_3yw_4z : xyz = v, |w_3+w_4| < p$, and w_3, w_4 are both substrings of $v\}$;
$\mathrm{M2} = \{xv : x \in \mathrm{NEG}(p,k,I)\}$;
$\mathrm{M3} = \{u_1u_3\$^{p^k}¢^d : (\exists u_2)u_1u_2u_3 = u \text{ and } 0 \le d < p^k\}$;
$\mathrm{M4} = \{(q^{p^k})^d z ¢^f : 0 \le d, f \le p^k+1, \text{ and } z \in \mathrm{NEG}(p,k,I)\}$.

LEMMA 8.3. *For all positive integers k, p, and all instances I of 1-in-3-SAT, any linear grammar G that is consistent with $POS^{lin}(p,k,I)$ and $NEG^{lin}(p,k,I)$ has size at least $p/3$.*

PROOF. Suppose to the contrary that G is a grammar consistent with $\mathrm{POS}^{\mathrm{lin}}(p,k,I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p,k,I)$ and the size of G is less than $p/3$. Then, by Proposition 2.6, G may be replaced with an equivalent thin grammar G' of size less than p. Without loss of generality, G' has at most p nonterminals; otherwise, some terminal(s) must be useless and could be discarded to obtain a smaller grammar.

Consider any derivation of the string v, which has length $2p^k$. Since the grammar is thin, each production generates at most one additional terminal; thus, there must be at least $2p^k$ applications of productions (steps) in the derivation. After each step (except the last) the sentential form thus far generated consists of some number of terminals, and exactly one nonterminal. Thus, one of the strictly less than p nonterminals appears in two different sentential forms less than p steps apart in the derivation of v. In other words, for some nonterminal A, we have

$$S \overset{*}{\Rightarrow} w_1Aw_2 \overset{*}{\Rightarrow} w_1w_3Aw_4w_2 \overset{*}{\Rightarrow} w_1w_3w_5w_4w_2 = v,$$

and such that $A \Rightarrow w_3Aw_4$ in less than p steps. By repeating this "subderivation" in the derivation of v, we obtain

$$S \overset{*}{\Rightarrow} w_1Aw_2 \overset{*}{\Rightarrow} w_1w_3Aw_4w_2 \overset{*}{\Rightarrow} w_1(w_3)^2A(w_4)^2w_2 \overset{*}{\Rightarrow} w_1(w_3)^2w_5(w_4)^2w_2 = v'.$$

Note that w_3 and w_4 are substrings of v such that $|w_3+w_4| < p$. Consequently, a negative example (type M1) is obtained by inserting these two strings anywhere in $v = w_1w_3w_5w_4w_2$. In particular, v' is a negative example generated by G', contradicting the fact that G' is consistent, completing the proof of Lemma 8.3. □

LEMMA 8.4. *Let I be an instance of 1-in-3-SAT. If τ is a solution of I, then for all positive integers k and p there is a linear grammar of size at most $6pn$ that is consistent with $POS^{lin}(p,k,I)$ and $NEG^{lin}(p,k,I)$.*

PROOF. We construct a linear grammar G of the desired size. First, convert the counter DFA $C(p,\tau)$ to a right-linear grammar G': Each of the p states becomes a nonterminal. For each of the n letters $a \in V$, and for any states T_1, T_2, the size 3 production $T_1 \to aT_2$ encodes the edge labeled with a from T_1 to T_2. For the nonterminal S, which corresponds to the start and final state of $C(p, \tau)$, add the production $S \to \lambda$. Then $L(G') = L(C(p, \tau))$ [15]. Finally, the grammar G may be obtained from G' by adding the production $S \to \$^p S ¢^p$.

The size of G' is exactly $3pn + 1$, and the size of G is exactly $3pn + 1 + 2p + 2$, which is at most $6pn$, since $n \geq 3$ for any instance of 1-in-3-SAT. G is consistent with $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$, since it clearly generates v, and, by construction, also generates xv for any string x accepted by $C(p, \tau)$. Since u is such a string, G generates uv also.

We argue that G is consistent with $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$. If G generates any string of M1, it must do so using only the two productions $S \to \$^p S ¢^p$ and $S \to \lambda$, since all other productions generate a character of $V - \{\$, ¢\}$. Clearly no element of M1 can be generated in this way, since all strings generated using these two productions must have length a multiple of p, which does not hold for any string of M1. Thus, G does not generate any string in M1.

If G generates some string $xv \in$ M2, then, by construction of G, we must have that G' generates x. But $L(G') = L(C(p, \tau))$, and thus $C(p, \tau)$ accepts the string $x \in \mathrm{NEG}(p, k, I)$, contradicting Lemma 4.2. Thus, G is consistent with M2.

G cannot generate any string of M3, since only strings with an equal number of \$ and ¢ symbols can be generated.

Finally, suppose that G generates some string $w = (q^{p^k})^d z ¢^f \in$ M4. Then $f = 0$, otherwise w contains an unequal number of \$ and ¢ symbols, and would not have been generated by G. It follows that G' must generate $w = (q^{p^k})^d z$ with $z \in \mathrm{NEG}(p, k, I)$. But $L(G') = L(C(p, \tau))$, and $C(p, \tau)$ does not accept w, since for all d, $(q^{p^k})^d$ leads from the start back to the start state, and thus if $C(p, \tau)$ accepted w, it would accept the string $z \in \mathrm{NEG}(p, k, I)$, contradicting Lemma 4.2. We conclude that G is consistent with all examples, completing the proof of Lemma 8.4. □

LEMMA 8.5. *Let k be any positive integer, and let p be a prime such that $p > 2^{k-1+m}$. If I is any instance of 1-in-3-SAT, and if G is a grammar of size less than $(p^k - 4)/6$ that is consistent with $POS^{lin}(p, k, I)$ and $NEG^{lin}(p, k, I)$, then I has some solution.*

Before proving Lemma 8.5, we present some supporting propositions. Throughout, we assume that the hypothesis of Lemma 8.5 is true.

Since $|G| < (p^k - 4)/6$, by Proposition 2.6 there exists a thin linear grammar G' that is consistent with $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$ and such that $|G'| \leq 3|G| < (p^k/2) - 2$. As in the previous reductions, we again concentrate on a particular positive example. All derivations used in the proof are with respect to the equivalent thin grammar G', instead of the original grammar G.

PROPOSITION 8.6. *The derivation of uv can be written as $S \Rightarrow \varphi C \psi$ and $C \Rightarrow \chi$, such that $\varphi \chi \psi = uv$, $|\chi| = p^k$, and ψ is a suffix of $¢^{p^k}$.*

PROOF. Since G' is thin, exactly one terminal is generated at each production step. Thus, there is some point in the derivation of uv such that exactly p^k terminals remain to be generated. Writing this as $S \overset{*}{\Rightarrow} \varphi C \psi$ and $C \overset{*}{\Rightarrow} \chi$, such that $\varphi\chi\psi = uv$, $|\chi| = p^k$, the proposition is proved if we show that ψ is a suffix of $¢^{p^k}$.

Assume to the contrary that ψ is not a suffix of $¢^{p^k}$, in which case $¢^{p^k}$ must be a proper suffix of ψ, that is, $|\psi| > p^k$. Since $|\chi| = p^k$, and there are p^k \$-symbols in uv, φ must be a prefix of u. Thus, all ¢-symbols are generated in the derivation of uv before the first \$-symbol is generated. Thus, the derivation of uv can be rewritten as:

$$S \overset{*}{\Rightarrow} \varphi_1 D_1 ¢ \overset{*}{\Rightarrow} \varphi_1\varphi_2 D_2 ¢^2 \overset{*}{\Rightarrow} \cdots \Rightarrow \varphi_1\varphi_2 \cdots \varphi_{p^k} D_{p^k} ¢^{p^k},$$

where $\varphi_1\varphi_2 \cdots \varphi_{p^k}$ is a prefix of u, $D_{p^k} \overset{*}{\Rightarrow} \omega\$^{p^k}$ and $\varphi_1\varphi_2 \cdots \varphi_{p^k}\omega\$^{p^k}¢^{p^k} = uv$.

Since $|G'| < (p^k/2) - 2$, G' has less than p^k nonterminals. Thus, there must exist $0 \le a < b \le p^k$ such that $D_a = D_b$, and we have that

$$S \overset{*}{\Rightarrow} \varphi_1\varphi_2 \cdots \varphi_a D_a ¢^a$$

and

$$D_b \overset{*}{\Rightarrow} \varphi_{b+1} \cdots \varphi_{p^k} D_{p^k} ¢^{p^k - b}.$$

Since $D_a = D_b$, we have

$$S \overset{*}{\Rightarrow} \varphi_1\varphi_2 \cdots \varphi_a\varphi_{b+1} \cdots \varphi_{p^k} D_{p^k} ¢^{p^k - b + a} \overset{*}{\Rightarrow} \varphi_1 \cdots \varphi_a\varphi_{b+1} \cdots \varphi_{p^k}\omega\$^{p^k}¢^{p^k - b + a}.$$

Let $u_1 = \varphi_1 \ldots \varphi_a$ (a prefix of u), let $u_3 = \varphi_{b+1} \cdots \varphi_{p^k}\omega$ (a suffix of u), and let $d = p^k - b + a$. Then $S \overset{*}{\Rightarrow} u_1u_3\$^{p^k}¢^d$, and for some u_2, $u_1u_2u_3 = u$. Thus, G' (and G) generates a negative example (type M3), contradicting its consistency. We conclude that ψ is a suffix of $¢^{p^k}$, completing the proof of Proposition 8.6. □

PROPOSITION 8.7. *There are numbers* $0 \le d$, $e \le p^k + 1$, *and* $0 \le f \le p^k$ *such that the derivation of the positive example uv can be written as*

$$S \overset{*}{\Rightarrow} \left(q^{p^k}\right)^d A ¢^f \overset{*}{\Rightarrow} \left(q^{p^k}\right)^d q^{p^k} \left(q^{p^k}\right)^e B ¢^f \overset{*}{\Rightarrow} uv.$$

PROOF. Proposition 8.6 ensures that in the derivation of uv, all of u is generated before any \$-symbol is generated to the right of the nonterminal, and thus we can rewrite the initial segment of the derivation as follows:

$$S = E_0\psi_0 \overset{*}{\Rightarrow} q^{p^k}E_1\psi_1 \overset{*}{\Rightarrow} \left(q^{p^k}\right)^2 E_2\psi_2 \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \left(q^{p^k}\right)^{p^k+1} E_{p^k+1}\psi_{p^k+1} \overset{*}{\Rightarrow} uv,$$

where $E_0 = S$, $\psi_0 = \lambda$ and all of the strings $\{\psi_i\}_{0 \le i \le p^k+1}$ consist of at most p^k ¢-symbols. Since there are $p^k + 2$ strings $\{\psi_i\}$, and only $p^k + 1$ values they may have, there exist a and b such that $0 \le a < b \le p^k + 1$ and $\psi_a = \psi_b$. Letting $A = E_a$, $d = a$, $¢^f = \psi_a = \psi_b$, $B = E_b$, and $e = b - a - 1$, the proposition follows. □

PROOF OF LEMMA 8.5. Proposition 8.7 ensures that in the derivation of the positive example uv, there is a nonterminal A such that $A \overset{*}{\Rightarrow} (q^{p^k})^{e+1}B$ for some $e \ge 0$. It follows that the subgrammar G'', with start symbol A, and with only those productions involved in the generation of $(q^{p^k})^{e+1}B$, is a right linear grammar.

Let G''' be the right linear grammar obtained by adding production $B \to \#$ to the right linear grammar G'', where $\# \notin V \cup \{\$, ¢\}$.

PROPOSITION 8.8. *Let $j = e + 1$. Then G''' is consistent with the examples $POS^j(p, k, I)\#$ and $NEG(p, k, I)\#$.*

Lemma 8.5 now follows from Proposition 8.8 by the following argument: Suppose the proposition is true. Note that the size of right linear grammar G''' satisfies $|G'''| = |G''| + 2 \leq |G'| + 2 < p^k/2$. Applying Lemma 2.5, we obtain an NFA A of size at most $2|G'''| < p^k$ that is consistent with $\mathrm{POS}^j(p, k, I)\#$ and $\mathrm{NEG}(p, k, I)\#$. By part 3 of Lemma 8.2, we conclude that I has a solution.

We now prove Proposition 8.8. G''' is consistent with $\mathrm{POS}^j(p, k, I)\#$ because (1) G''' has start symbol A, (2) by definition, we have included productions such that $A \overset{*}{\Rightarrow} (q^{p^k})^{e+1}B$, (3) $j = e + 1$, and (4) $B \to \#$ is an additional production of G'''.

Now suppose that G''' generates some element of $\mathrm{NEG}(p, k, I)\#$. Then clearly, $A \overset{*}{\Rightarrow} z\#$, where $z \in \mathrm{NEG}(p, k, I)$, and thus in grammar G'' and G' we have $A \overset{*}{\Rightarrow} z$.

From Proposition 8.7, in grammar G', we have

$$S \Rightarrow \left(q^{p^k}\right)^d A ¢^f \overset{*}{\Rightarrow} \left(q^{p^k}\right)^d z ¢^f,$$

which is a negative example (type M4), contradicting the consistency of G'. Thus, G''' is consistent with $\mathrm{POS}^j(p, k, I)\#$ and $\mathrm{NEG}(p, k, I)\#$, completing the proof of Proposition 8.8 and Lemma 8.5. □

PROOF OF THEOREM 8.1. Suppose $f(opt)$ is bounded above by some polynomial in opt. Then there exists a number k such that for all sufficiently large values of opt,

$$f(opt) \leq \frac{1}{6}\left(\frac{opt}{12}\right)^{k/2} - \frac{2}{3}.$$

Consequently, to show that MIN-CON(LIN, LIN) is not $f(opt)$-approximable unless P = NP, it suffices to show for all k that MIN-CON(LIN, LIN) is not $(1/6(opt/12)^{k/2} - 2/3)$-approximable unless P = NP.

As in the proof of Theorem 6.1, for each k we define a polynomial-time transformation T_k, that on input instance I of a 1-in-3-SAT problem with n variables, determines which reduction $R^{\mathrm{lin}}_{p,k}(I)$ to apply. More specifically, T_k first determines whether $n > 2^{k-1+m}$. If not, then T_k halts and outputs nothing. Otherwise, T_k finds the smallest prime number p satisfying $2^{k-1+m} < n \leq p \leq 2n$. After obtaining such a prime p, T_k then computes and outputs $R^{\mathrm{lin}}_{p,k}(I)$. By the same argument as that given in the proof of Theorem 6.1, for each constant k, T_k is computable in time polynomial in $|I|$.

Let k be given. Suppose, contrary to what we must show, that there exists a constant c and a polynomial-time algorithm APPROX such that for all instances of MIN-CON(LIN, LIN) with optimal solution satisfying $opt \geq c$, APPROX is guaranteed to output a consistent linear grammar of size less than

$$\frac{1}{6}\left(\frac{opt}{12}\right)^{k/2} - \frac{2}{3}.$$

As in the proof of Theorem 6.1, we construct a polynomial-time algorithm DECIDE for determining whether instances of 1-in-3-SAT have a solution.

On input instance I with n variables, DECIDE determines if $n > \max\{3c, 2^{k-1+m}\}$. If not then DECIDE determines whether I has a solution by trying all possible assignments (a constant number, since c, k, and m are constants). Otherwise, DECIDE computes $T_k(I)$, and gives the result as input to subroutine APPROX. If APPROX returns a linear grammar G of size $|G| < (n^k - 4)/6$, then DECIDE outputs "I has a solution." Otherwise, DECIDE outputs, "I does not have a solution."

Clearly DECIDE runs in time polynomial in $|I|$, since both T_k and APPROX run in polynomial time. We must show that DECIDE is correct. Certainly DECIDE is correct for all instances I with $n \le \max\{3c, 2^{k-1+m}\}$ variables, since these decisions are made by exhaustive search.

Suppose that I is an instance with $n > \max\{3c, 2^{k-1+m}\}$ variables, and that DECIDE outputs "I has a solution." Then $T_k(I)$ gives the MIN-CON(LIN, LIN) instance $R^{\mathrm{lin}}_{p,k}(I)$ as input to APPROX (where p is the smallest prime between n and $2n$). Since APPROX returns a linear grammar G of size $|G| < (n^k - 4)/6 \le (p^k - 4)/6$, it follows by Lemma 8.5 that I has a solution.

Conversely, suppose that I is an instance with $n > \max\{3c, 2^{k-1+m}\}$ variables, and I has a solution. Since n is large enough, DECIDE runs $T_k(I)$ and obtains the instance (of MIN-CON(LIN, LIN)) $R^{\mathrm{lin}}_{p,k}(I)$ consisting of sets $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$, where p is the smallest prime between n and $2n$.

By Lemmas 8.3 and 8.4, the size *opt* of the smallest linear grammar consistent with $\mathrm{POS}^{\mathrm{lin}}(p, k, I)$ and $\mathrm{NEG}^{\mathrm{lin}}(p, k, I)$ satisfies $p/3 \le \mathit{opt} \le 6pn$. Since $3c < n \le p$, we have

$$c < \frac{n}{3} \le \frac{p}{3} \le \mathit{opt},$$

and *opt* is large enough so that the bound on the performance of APPROX must apply, thus APPROX must find a consistent linear grammar G such that

$$|G| < \frac{1}{6}\left(\frac{\mathit{opt}}{12}\right)^{k/2} - \frac{2}{3}.$$

Since $\mathit{opt} \le 6pn$ and $p \le 2n$, we have $\mathit{opt} \le 12n^2$. Consequently, the consistent linear grammar G returned by APPROX satisfies

$$|G| < \frac{1}{6}\left(\frac{12n^2}{12}\right)^{k/2} - \frac{2}{3} \le \frac{n^k}{6} - \frac{2}{3} = \frac{n^k - 4}{6},$$

and DECIDE outputs "I has a solution." □

9. *Approximability and Learnability*

Following [10], let Π be a minimization problem, let I be any instance of Π, and let A be an approximation algorithm. There are a number of ways we might wish to measure the performance of A. One measure is to simply express the size of the approximate solution as a function of the size of the smallest feasible solution. Consider the minimization problem MIN-CON $(\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}})$ of Theorem 6.1. In this case, an instance I consists of a collection of positive and negative examples, $\mathit{opt}(I)$ is the number of states in the smallest consistent DFA, and for any approximation algorithm A, $A(I)$ is the consistent NFA produced by A and $|A(I)|$ is the number of states of the NFA produced. Theorem 6.1 states that if $\mathrm{P} \neq \mathrm{NP}$, then no polynomial-time algorithm A for MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$) can achieve $|A(I)| < opt(I)^k$ for any constant k.

Another reasonable measure is to express the ratio $|A(I)|/opt(I)$ as a function of the size of the input (denoted by $|I|$). In the case of MIN-CON, $|I|$ is the total number of letters in all of the examples. Using the quadratic nonapproximability result of Section 3, we prove a lower bound on the ratio, given below in Theorem 9.1. Similar bounds could be achieved with the polynomial nonapproximability results of Section 6, however, the lower bound obtained would be much lower. Note that if $\mathrm{P} = \mathrm{NP}$, then $|A(I)|/opt(I) = 1$ for MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$). For an arbitrary DFA or NFA M, let $\|M\|$ be the number of bits needed to encode M according to some standard encoding. Recall that $|M|$ is the number of states of M. We assume that $\|M\| \geq |M|$.

THEOREM 9.1. *If $P \neq NP$, then for any $\epsilon > 0$, for any polynomial-time approximation algorithm A for MIN-CON($DFA^{\{0,1\}}, NFA^{\{0,1\}}$), and for infinitely many positive integers c, there are instances I of MIN-CON($DFA^{\{0,1\}}, NFA^{\{0,1\}}$) such that $opt(I) \geq c$, and for which the performance ratio of A satisfies*

$$\frac{\|A(I)\|}{opt(I)} \geq \frac{|A(I)|}{opt(I)} > |I|^{1/(14+\epsilon)}.$$

PROOF. The first inequality holds by our earlier assumption that $\|M\| \geq |M|$. To obtain the second inequality, we convert the reduction of Section 3 (which was used to show the quadratic nonapproximability result) to the two-letter case. A similar conversion was done in Section 6 for the polynomial nonapproximability result of Section 4.

For any odd number p, let $R_p^{\{0,1\}}$ map any instance I' of 1-in-3-SAT to an instance I of MIN-CON($\mathrm{DFA}^{\{0,1\}}, \mathrm{NFA}^{\{0,1\}}$), which consists of the two-letter examples $\widehat{\mathrm{POS}}(p, I')$ and $\widehat{\mathrm{NEG}}(p, I')$. By Proposition 3.6, and from the fact that $\hat{v}_x$ is a $\log n$ length bit string, it follows that the total length of all examples of I is at most $cp^{14}n^4 \log n$, for some constant c. Also, if I' has a solution, then $opt(I) \leq pn$, since for any solution τ of I' there is a pn state DFA for $L(\hat{C}(p, \tau))$ (from the proof of Lemma 6.3). If there is a consistent NFA M with less than p^2 states then I' has a solution. Then an approximation algorithm that guarantees

$$\frac{|A(I)|}{opt(I)} < \frac{p}{n}$$

can be used to solve an NP-complete problem as follows: If I' has a solution, then

$$\frac{|A(I)|}{pn} \leq \frac{|A(I)|}{opt(I)} < \frac{p}{n},$$

which implies that $|A(I)| < p^2$; on the other hand, if I' does not have a solution, then $|A(I)| \geq p^2$. We conclude that (unless $\mathrm{P} = \mathrm{NP}$), no polynomial-time approximation algorithm can guarantee

$$\frac{|A(I)|}{opt(I)} < \frac{p}{n}.$$

To complete the proof of the theorem, it suffices to show that the guarantee

$$\frac{|A(I)|}{opt(I)} \leq |I|^{1/(14+\epsilon)}$$

implies that

$$\frac{|A(I)|}{opt(I)} < \frac{p}{n}$$

for some value of p that is polynomially related to n. By our analysis above concerning $|I|$, it suffices to show that the inequality

$$(cp^{14}n^4 \log n)^{1/(14+\epsilon)} < \frac{p}{n} \tag{1}$$

has a solution for p that is at most polynomially larger than n.

To see this, let $p = n^r$, and solve for r. Inequality (1) is equivalent to

$$r > \frac{\log_n(c) + \log_n(\log n) + 18 + \epsilon}{\epsilon}.$$

Let

$$r' = \frac{c + 19 + \epsilon}{\epsilon}$$

and let p be the smallest odd integer larger than $n^{r'}$. Then, inequality (1) is satisfied, and since r' is a constant, p is clearly polynomial in $|I'|$ and $|I|$. □

A standard measure of performance of an approximation algorithm A is the *asymptotic performance ratio* (denoted R_A^∞) [10], defined by $R_A^\infty = \inf\{r \geq 1$: for some positive integer k,

$$\frac{|A(I)|}{opt(I)} \leq r$$

for all instances I such that $opt(I) \geq k\}$. Theorem 9.1 above implies that, unless P = NP, R_A^∞ is infinite for MIN-CON(DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$).

We next introduce another measure of approximation performance which is motivated by recent work in computational learning theory. Pac-learning of a class of objects (e.g., of DFAs) requires that from randomly generated examples of some unknown member of the class (the target DFA), a (possibly different) member of the *same class* (called the hypothesis) is produced that is likely to agree (in a precisely quantified way) with future examples generated from the same distribution [25]. A relaxation of this definition allows pac-learning of a class *in terms of* some other class [22]. For example, to pac-learn DFAs in terms of NFAs, a learning algorithm may choose its hypothesis from the class of NFAs. Thus pac-learning of DFAs in terms of NFAs is easier than pac-learning of DFAs (in terms of DFAs). It follows from [6] that the pac-learnability of DFAs in terms of NFAs would be established from the existence of a polynomial-time algorithm A, and any constants $\alpha \geq 0$ and $\beta < 1$, with the following properties: A, on input of any instance I of the MIN-CON($\text{DFA}^{\{0,1\}}$, $\text{NFA}^{\{0,1\}}$) problem (i.e., finite sets POS and NEG), will produce a consistent NFA $A(I)$ such that

$$\| A(I) \| \leq opt(I)^{\alpha} card(I)^{\beta}, \tag{2}$$

where $opt(I)$ is the size of the smallest DFA consistent with the examples of I, and $card(I)$ is the number of examples of I. Further, it has been shown that the existence of an approximation algorithm that produces a DFA whose size meets the above bound is *equivalent* to the existence of a learning algorithm for DFAs (in terms of DFAs) [7].

If we restrict our attention to pac-learning DFAs in terms of NFAs from *polynomially length bounded* examples (all examples with nonzero probability are at most polynomially larger than the size of the DFA to be learned), then the results of [6] also give that pac-learnability is implied by the existence of a polynomial-time algorithm A, and any constants $\alpha \geq 0$ and $\beta < 1$, such that on input of any instance I of MIN-CON($\text{DFA}^{\{0,1\}}$, $\text{NFA}^{\{0,1\}}$), A outputs a consistent NFA $A(I)$ such that

$$\| A(I) \| \leq opt(I)^{\alpha} |I|^{\beta}, \tag{3}$$

where $|I|$ is the total size of all examples of I. (Again, for the case where the learning algorithm must output a DFA and the examples are polynomially length bounded, the above sufficient condition for pac-learning DFAs has also been shown to be necessary [7].)

Consequently, an approach toward pac-learning of DFAs in terms of NFAs from polynomially length bounded examples is to produce a polynomial-time algorithm that satisfies the guarantee of inequality (3) for instances of MIN-CON($\text{DFA}^{\{0,1\}}$, $\text{NFA}^{\{0,1\}}$) for which the elements of POS and NEG are polynomially length bounded in the size of the smallest consistent DFA.

All of the nonapproximability results of this paper in fact apply to a restricted version of MIN-CON in which the size of the elements of POS and NEG are polynomially length bounded. This can be seen as follows: In each reduction, the length of the longest example is polynomial in p (a parameter of the reduction) and n (the number of variables of the instance of 1-in-3-SAT). In all reductions, $p \leq c \cdot opt$, for some constant c, and thus if p is chosen at least as large as n then the length of the longest example is polynomially length bounded in opt. We complete this section by investigating the implications that these (and other) reductions have with respect to the performance criterion given by inequality (3).

Recently, Kearns and Valiant [16] have shown that DFAs are not *polynomially predictable* based on any of several well established cryptographic assumptions: that deciding quadratic residuosity is hard, that the RSA public key cryptosystem is secure, or that factoring Blum integers is hard. Their results hold even if the examples are polynomially length bounded.

Polynomial predictability is equivalent to pac-learnability in terms of the class of polynomially sized programs (i.e., the hypothesis may be any polynomial-time algorithm for classification of examples which is representable with polynomially many bits) [13]. In [16], it is shown that the nonpredictability of DFAs, together with the results in [6], imply that there is no polynomial-time algorithm A for MIN-CON(DFA$^{\{0,1\}}$, NFA$^{\{0,1\}}$) producing a consistent NFA $A(I)$ such that inequality (3) holds for any constants $\alpha \geq 0$ and $\beta < 1$, unless the previously mentioned cryptographic assumptions are false.

Our results, given in Corollary 9.2, complement those presented in [16]; we obtain analogous nonapproximability results for restricted choices of α and β, but using the (ostensibly weaker) assumption that P $\neq$ NP.

COROLLARY 9.2. *If $P \neq NP$, then for any polynomial-time approximation algorithm A, and for infinitely many positive integers c, there are instances I of MIN-CON($DFA^{\{0,1\}}$, $NFA^{\{0,1\}}$) such that $opt(I) \geq c$, and for which the performance of A satisfies $\|A(I)\| \geq |A(I)| > opt(I)^{\alpha}|I|^{\beta}$ for any of the following choices of α and β:*

(1) *for any $\alpha \geq 0$, when $\beta = 0$;*
(2) *for $\alpha = 1$ and any $\beta < 1/14$;*
(3) *for any $0 \leq \alpha = 1 + \alpha' < 2$ and any $\beta < (1 - \alpha')/14$.*

PROOF. The inequality $\|A(I)\| \geq |A(I)|$ follows by assumption on the size measures. Observe that the inequality $|A(I)| \leq opt(I)^{1+\alpha'}|I|^{\beta}$ is equivalent to $|A(I)|/opt \leq opt^{\alpha'}|I|^{\beta}$. Case 1 follows from Theorem 6.1. Case 2 is equivalent to the statement of Theorem 9.1, where $\beta = 1/(14 + \epsilon)$ has been rewritten as $\beta < 1/14$. Case 3 follows by an argument similar to the proof of Theorem 9.1. The only significant modification is that inequality (1), which addresses the case $\alpha' = 0$, is replaced by

$$opt(I)^{\alpha'}|I|^{(1-\alpha')/(14+\epsilon)} \leq (pn)^{\alpha'}(cp^{14}n^4 \log n)^{(1-\alpha')/(14+\epsilon)} < \frac{p}{n},$$

which is equivalent to

$$(cp^{14}n^4 \log n)^{1/(14+\epsilon)} < \frac{p}{n^{(1+\alpha')/(1-\alpha')}}.$$

Set the constant

$$r' = \frac{c + 5 + 14\gamma + \epsilon\gamma}{\epsilon},$$

where

$$\gamma = \frac{1 + \alpha'}{1 - \alpha'}.$$

Then the first odd number $p \geq n^{r'}$ makes the previous inequality true. □

Besides the use of different assumptions (i.e., cryptographic versus P $\neq$ NP), another difference between our work and that appearing in [16], is that while the cryptographic based results of [16] rely on the inability to predict DFAs, the subfamily of DFAs for which we show nonapproximability results is actually easy to predict. The class of CDFAs accept *permutation invariant* languages (w is accepted iff any word formed by permuting the characters in w is accepted), and for each CDFA the start state equals the unique final state. DFAs with these properties have been shown to be predictable [14], thus the techniques of

[16] cannot apply to show that the related MIN-CON problem for this restricted class of DFAs is not polynomially approximable.

10. *Conclusion*

The problem of finding an approximately small DFA consistent with a finite sample was investigated. It was shown that unless P = NP, no polynomial-time algorithm can be guaranteed to produce a DFA, NFA, regular expression, or regular grammar of size at most polynomially larger than the smallest consistent DFA. The minimum consistent linear grammar problem was also shown to have similar nonapproximability properties. Our results are summarized by Theorems 3.1, 4.1, 5.1, 6.1, 7.1, 8.1, 9.1, and their corollaries.

It should be noted that the proofs of each of these theorems was nonconstructive in the sense that the existence of an approximately small NFA (regular grammar, regular expression, linear grammar, respectively), implies the existence of a solution to an instance of 1-in-3-SAT. The proofs can be easily modified so as to be constructive, that is, so that from an approximately small NFA (for example), a solution to the relevant 1-in-3-SAT instance can be found in polynomial time. Of course, since the problem of finding a solution to a 1-in-3-SAT problem reduces in polynomial time to the decision problem, our observation concerning constructiveness is of dubious interest.

In our definition of approximability (Definition 2.9) we required that the approximation algorithm must output a representation of size less than the upper bound. It should be noted that all nonapproximability results of this paper still hold when the approximation algorithm only *decides* whether there exists a consistent representation of size less than the upper bound.

Because the DFAs used in the reductions of this paper were of a very special form (CDFAs or counter-like DFAs), the proof of Theorem 4.1 (for example) actually shows the stronger result that for any constant k, MIN-CON(CDFA, NFA) (and thus MIN-CON(DFA, NFA)) is not opt^k-approximable unless P = NP. As discussed at the end of the previous section, it has been shown that CDFAs are polynomially predictable [14]. (See [1] for additional results on the prediction of classes of commutative languages.)

The research presented here suggests a large number of open problems. The investigation of the approximability of versions of the MIN-CON problem other than the ones considered here seems appropriate. Can the nonapproximability results (assuming P $\neq$ NP) for MIN-CON(LIN, LIN) be extended to MIN-CON(CFG, CFG)? (At present, it is not even known whether it is NP-hard to find the *smallest* consistent CFG). In the problem MIN-CON(DFA, NFA), the approximation algorithm has the "freedom" to output a consistent NFA instead of a consistent DFA. Further generalizing along these lines, it would be of interest to know whether similar nonapproximability results may be shown for MIN-CON(DFA, 2DFA), MIN-CON(DFA, 2NFA), MIN-CON(DFA, LIN), and MIN-CON(DFA, CFG), etc., where 2DFAs and 2NFAs are the two-way versions of DFAs and NFAs, respectively. It has been shown that MIN-CON(DNF, DNF) is not $(2-\epsilon)opt$-approximable [22], (where DNF denotes the set of Boolean formulas in disjunctive normal form). Can this result be strengthened using an adaptation of the methods used here?

Another set of Boolean functions other than DNF that is of interest in computational learning theory is the set of Boolean decision trees (DT). It has been recently shown in [12] that MIN-CON(DT, DT) is not $opt + opt^{\beta}$ approximable for any constant $\beta < 1$. The decision trees used in the reduction are very unbalanced. Let a *balanced decision tree* (BDT) have the additional property that all leaves are on the same level. Can it be decided in polynomial time whether there is a BDT that is consistent with given examples POS and NEG?

In Section 9, the work of [16] was discussed. These results show nonapproximability for DFAs based on cryptographic assumptions. By relying instead on the assumption that $P \neq NP$, our results strengthen theirs, but only for a subrange of the parameters α and β. Can the entire range of results presented in [16] be proven using only the assumption that $P \neq NP$?

Angluin showed in [3] that DFAs are learnable in polynomial time if the learning algorithm is allowed equivalence and membership queries with respect to a fixed unknown target DFA. An equivalence query consists of a hypothesized DFA, and the response (of the teacher) is either "correct, the hypothesized DFA and the target DFA are equivalent," or else a counterexample word is returned on which the queried DFA and the target DFA disagree. A membership query is a word w, and the response of the teacher is "yes" iff w is accepted by the target DFA.

Angluin's algorithm (call it A) can be used to construct an algorithm B that solves MIN-CON(DFA, DFA) in polynomial time making use of the following type of query, which we call a *consistency query*: A consistency query consists of a pair (EX, w), where EX is a finite set of labeled examples (each example is a word, and the label is either "+" or "−" indicating whether the word is a positive or negative example), and w is an additional (unlabeled) word. The response to the query (EX, w) is "+" if the size of a smallest DFA consistent with EX and additional positive example w is the same as the size of a smallest DFA consistent with the examples EX. Otherwise, the response is "−".

The algorithm B for MIN-CON(DFA, DFA) using consistency queries behaves as follows. B receives examples POS and NEG and is to output a consistent DFA with the minimum number of states. B initializes EX to POS ∪ NEG (labeling all elements of POS with "+" and all of NEG with "−"), and simulates A. When A asks a membership query "w", B asks a consistency query using EX as the example set, and w as the additional word. Each consistency query produces a new labeled example which is then added to EX. When A asks an equivalence query with DFA M as an argument, B checks if M is consistent with POS and NEG. If M is consistent, then B outputs M and stops, otherwise, B returns to A an inconsistent example of POS ∪ NEG (the counterexample to A's equivalence query) and continues the simulation of A.

The properties of Angluin's algorithm guarantee that after polynomially many consistency queries (polynomial in the size of POS, NEG, and the value *opt*) the machine output by B has *opt* states.

An interesting problem is to determine lower bounds on the number of consistency queries required to produce the smallest consistent DFA (or a DFA of size at most opt^k, where *opt* is the smallest). Clearly, more than a logarithmic number of consistency queries are necessary, since a polynomial-time algorithm could try all possible answers to these queries.

This paper presented a number of very strong nonapproximability proofs for certain types of NP-hard optimization problems. A final open problem is whether similar proof techniques can be used to obtain nonapproximability results for other classical NP-hard problems.

ACKNOWLEDGMENTS. We thank Dana Angluin, Ming Li, and Umesh Vazirani for sparking our interest in this problem, Don Hatch for helping to clarify the group theoretic structure in some of our early reductions, and Michael Kearns for discussions regarding the material of Section 9. Additional thanks to Dana for carefully plowing through an early draft containing the technical arguments (and little else). Finally, we thank the referees for the time and effort that they put into the review of this manuscript.

REFERENCES

1. ABE, N. Learning commutative deterministic finite state automata in polynomial time. In *Proceedings of the 1st International Workshop on Algorithmic Learning Theory*, Japanese Society for Artificial Intelligence, Tokyo, Japan, 1990, pp. 223–235.
2. ANGLUIN, D. An application of the theory of computational complexity to the study of inductive inference. Ph.D. dissertation, Electrical Engineering and Computer Science Department, University of California at Berkeley, Berkeley, Calif., 1976.
3. ANGLUIN, D. Learning regular sets from queries and counterexamples. *Inf. Comput. 75*, 2 (1987), 87–106.
4. ANGLUIN, D. Negative results for equivalence queries. *Mach. Learn. 5*, 2 (1990), 121–150.
5. ANGLUIN, D. On the complexity of minimum inference of regular sets. *Inf. Cont. 39*, 3 (1978), 337–350.
6. BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., AND WARMUTH, M. Occam's razor. *Inf. Proc. Lett. 24* (1987), 377–380.
7. BOARD, R., AND PITT, L. On the necessity of Occam algorithms. *Theoret. Comput. Sci. 100* (1992), 157–184.
8. EHRENFEUCHT, A., AND ZEIGER, P. Complexity measures for regular expressions. *J. Comput. Syst. Sci. 12*, 2 (1976), 134–146.
9. GAREY, M., AND JOHNSON, D. The complexity of near-optimal graph coloring. *J ACM 23*, 1 (June 1976), 43–49.
10. GAREY, M., AND JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, Calif., 1979.
11. GOLD, E. M. Complexity of automaton identification from given data. *Inf Control 37* (1978), 302–320.
12. HANCOCK, T. On the difficulty of finding small consistent decision trees. Unpublished manuscript, Aiken Computation Laboratory, Harvard Univ., Cambridge, Mass., 1989.
13. HAUSSLER, D., KEARNS, M., LITTLESTONE, N., AND WARMUTH, M. K. Equivalence of models for polynomial learnability. *Inf. Comput. 95* (1991), 129–161.
14. HELMBOLD, D., SLOAN, R., AND WARMUTH, M. K. Learning integer lattices. *SIAM J. Comput. 21*, 2 (1992), 240–266.
15. HOPCROFT, J. E., AND ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 1979.
16. KEARNS, M., AND VALIANT, L. G. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (Seattle, Wash., May 15–17). ACM, New York, 1989, pp. 433–444.
17. KOLAITIS, PH. G., AND THAKUR, M. N. Approximation properties of NP minimization problems. In *Proceedings of the 6th Annual IEEE Conference on Structure in Complexity Theory.* IEEE Computer Society Press, Washington, D.C., 1991, pp. 353–366.
18. LI, M., AND VAZIRANI, U. On the learnability of finite automata. In *Proceedings of the 1988 Workshop on Computational Learning Theory*. Morgan-Kaufmann, San Mateo, Calif., 1988, pp. 359–370.
19. NIVEN, I., AND ZUCKERMAN, H. S. *An Introduction to the Theory of Numbers*. Wiley, New York, 1972.
20. PANCONESI, A., AND RANJAN, D. Quantifiers and approximation. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, (Baltimore, Md., May 14–16). ACM, New York, 1990, pp. 446–456.

21. PAPADIMITRIOU, C. H., AND YANNAKAKIS, M. Optimization, approximation, and complexity classes. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, Ill., May 2–4). ACM, New York, 1988, pp. 229–234.
22. PITT, L., AND VALIANT, L. G. Computational limitations on learning from examples. *J. ACM 35*, 4 (Oct. 1988), 965–984.
23. SCHAEFER, T. J. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing* (San Diego, Calif., May 1–3). ACM, New York, 1978, pp. 216–226.
24. TRAKHTENBROT, B. A., AND BARZDIN, YA. M. *Finite Automata*. North-Holland, Amsterdam, 1973, pp. 98–99.
25. VALIANT, L. G. A theory of the learnable. *Commun. ACM 27*, 11 (Nov 1984), 1134–1142.

RECEIVED FEBRUARY 1989; REVISED JULY 1991; ACCEPTED AUGUST 1991