Finite Automata, Palindromes, Powers, and Patterns

Terry Anderson, Narad Rampersad, Nicolae Santean[⋆], and Jeffrey Shallit

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1, Canada
tanderson@uwaterloo.ca, nrampersad@cs.uwaterloo.ca
nsantean@iusb.edu, shallit@graceland.uwaterloo.ca

Abstract. Given a language L and a nondeterministic finite automaton M, we consider whether we can determine efficiently (in the size of M) if M accepts at least one word in L, or infinitely many words. Given that M accepts at least one word in L, we consider how long the shortest word can be. The languages L that we examine include the palindromes, the non-palindromes, the k-powers, the non-k-powers, the powers, the non-powers (also called primitive words), and words matching a general pattern.

1 Introduction

Let $L \subseteq \Sigma^*$ be a fixed language, and let M be a deterministic finite automaton (DFA) or nondeterministic finite automaton (NFA) with input alphabet Σ. In this paper we are interested in three questions:

1. Whether we can efficiently decide (in terms of the size of M) if $L(M)$ contains at least one element of L, that is, if $L(M) \cap L \neq \emptyset$;
2. Whether we can efficiently decide if $L(M)$ contains infinitely many elements of L, that is, if $L(M) \cap L$ is infinite;
3. Given that $L(M)$ contains at least one element of L, what is a good upper bound on the shortest element of $L(M) \cap L$?

As an example, consider the case where $\Sigma = \{\mathtt{a}\}$, L is the set of primes written in unary, that is, $\{\mathtt{a}^i \ : \ i \text{ is prime } \}$, and M is a NFA with n states.

To answer questions (1) and (2), we first rewrite M in Chrobak normal form [5]. Chrobak normal form consists of an NFA M' with a "tail" of $O(n^2)$ states, followed by a single nondeterministic choice to a set of disjoint cycles containing at most n states. Computing this normal form can be achieved in $O(n^5)$ steps by a result of Martinez [17].

Now we examine each of the cycles produced by this transformation. Each cycle accepts a finite union of sets of the form $(\mathtt{a}^t)^*\mathtt{a}^c$, where t is the size of

[⋆] Author's current address: Department of Computer and Information Sciences, Indiana University South Bend, 1700 Mishawaka Ave., P.O. Box 7111, South Bend, IN 46634, USA.

the cycle and $c \leq n^2 + n$; both t and c are given explicitly from M'. Now, by Dirichlet's theorem on primes in arithmetic progressions, $\gcd(t, c) = 1$ for at least one pair (t, c) induced by M' if and only if M accepts infinitely many elements of L. This can be checked in $O(n^2)$ steps, and so we get a solution to question (2) in polynomial time.

Question (1) requires a little more work. From our answer to question (2), we may assume that $\gcd(t, c) > 1$ for all pairs (t, c), for otherwise M accepts infinitely many elements of L and hence at least one element. Each element in such a set is of length $kt + c$ for some $k \geq 0$. Let $d = \gcd(t, c) \geq 2$. Then $kt + c = (kt/d + c/d)d$. If $k > 1$, this quantity is at least $2d$ and hence composite. Thus it suffices to check the primality of c and $t + c$, both of which are at most $n^2 + 2n$. We can precompute the primes $< n^2 + 2n$ in linear time using a modification of the sieve of Eratosthenes [18], and check if any of them are accepted. This gives a solution to question (1) in polynomial time.

On the other hand, answering question (3) essentially amounts to estimating the size of the least prime in an arithmetic progression, an extremely difficult question that is still not fully resolved [9], although it is known that there is a polynomial upper bound.

Thus we see that asking these questions, even for relatively simple languages L, can quickly take us to the limits of what is known in formal languages and number theory.

In this paper we examine questions (1)-(3) in the case where M is an NFA and L is either the set of palindromes, the set of k-powers, the set of powers, the set of words matching a general pattern, or their complements.

In some of these cases, there is previous work. For example, Ito et al. [12] studied several circumstances in which primitive words (non-powers) may appear in regular languages. As a typical result in [12], we mention: "A DFA over an alphabet of 2 or more letters accepts a primitive word iff it accepts one of length $\leq 3n-3$, where n is the number of states of the DFA". Horváth, Karhumäki and Kleijn [11] addressed the decidability problem of whether a language accepted by an NFA is palindromic (i.e., every element is a palindrome). They showed that the language accepted by an NFA with n states is palindromic if and only if all its words of length shorter than $3n$ are palindromes.

A preliminary version of the full version of this paper is available online [2].

2 Notions and Notation

Let Σ be an alphabet, i.e., a nonempty, finite set of symbols (letters). By Σ^* we denote the set of all finite words over Σ, and by ε, the empty word. For $w \in \Sigma^*$, we denote by w^R the word obtained by reversing the order of symbols in w. A *palindrome* is a word w such that $w = w^R$. If L is a language over Σ, i.e., $L \subseteq \Sigma^*$, we say that L is *palindromic* if every word $w \in L$ is a palindrome.

Let $k \geq 2$ be an integer. A word y is a *k-power* if y can be written as $y = x^k$ for some non-empty word x. If y cannot be so written for any $k \geq 2$, then y is *primitive*. A 2-power is typically referred to as a *square*, and a 3-power as a *cube*.

Patterns are a generalization of powers. A *pattern* is a non-empty word p over a *pattern alphabet* Δ. The letters of Δ are called *variables.* A pattern p *matches* a word $w \in \Sigma^*$ if there exists a non-erasing morphism $h : \Delta^* \rightarrow \Sigma^*$ such that $h(p) = w$. Thus, a word w is a k-power if it matches the pattern a^k.

We define an NFA (or DFA) as the usual 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$. The size of M is the total number N of its states and transitions. When we want to emphasize the components of M, we say M has n states and t transitions, and define $N := n + t$.

We note that if M is an NFA or NFA-ϵ, we can remove all states that either cannot be reached from the start state or cannot reach a final state (the latter are called *dead states*) in linear time (in the number of states and transitions) using depth-first search. We observe that $L(M) \neq \emptyset$ if and only if any states remain after this process, which can be tested in linear time. Similarly, if M is a NFA, then $L(M)$ is infinite if and only if the corresponding digraph has a directed cycle. This can also be tested in linear time.

We will also need the following well-known results [10]:

Theorem 1. *Let M be an NFA with n states. Then*

(a) $L(M) \neq \emptyset$ if and only if M accepts a word of length $< n$.
(b) $L(M)$ is infinite if and only if M accepts a word of length ℓ, $n \leq \ell < 2n$.

A language L is called *slender* if there is a constant C such that, for all $n \geq 0$, the number of words of length n in L is less than C. The following characterization of slender regular languages has been independently rediscovered several times in the past [14,24,19].

Theorem 2. *Let $L \subseteq \Sigma^*$ be a regular language. Then L is slender if and only if it can be written as a finite union of languages of the form uv^*w, where $u, v, w \in \Sigma^*$.*

For further background on finite automata and regular languages we refer the reader to Yu [26].

3 Testing If an NFA Accepts at Least One Palindrome

Over a unary alphabet, every string is a palindrome, so problems (1)-(3) become trivial. Let us assume, then, that the alphabet Σ contains at least two letters. Although the palindromes over such an alphabet are not regular, the language

$$L' = \{x \in \Sigma^* \ : \ xx^R \in L(M) \text{ or there exists } a \in \Sigma \text{ such that } xax^R \in L(M)\}$$

is, in fact, regular, as often shown in a beginning course in formal languages [10, p. 72, Exercise 3.4 (h)]. We can take advantage of this as follows:

Lemma 1. *Let M be an NFA with n states and t transitions. Then there exists an NFA M' with $n^2 + 1$ states and $\leq 2t^2$ transitions such that $L(M') = L'$.*

Corollary 1. *Given an NFA M with n states and t transitions, we can determine if M accepts a palindrome in $O(n^2 + t^2)$ time.*

Corollary 2. *Given an NFA M, we can determine if $L(M)$ contains infinitely many palindromes in quadratic time.*

Corollary 3. *If an NFA M accepts at least one palindrome, it accepts a palindrome of length $\leq 2n^2 - 1$.*

Rosaz [21] also gave a proof of this last corollary. The quadratic bound is tight, up to a multiplicative constant, in the case of alphabets with at least two letters, and even for DFAs:

Proposition 1. *For infinitely many n there exists a DFA M_n with n states over a 2-letter alphabet such that the shortest palindrome accepted by M_n is of length $\geq n^2/2 - 3n + 5$.*

4 Testing If an NFA Accepts at Least One Non-palindrome

In this section we consider the problem of deciding if an NFA accepts at least one non-palindrome. Equivalently, we consider the problem: *Given an NFA M, is $L(M)$ palindromic?*

Again, the problem is trivial for a unary alphabet, so we assume $|\Sigma| \geq 2$. Horváth, Karhumäki, and Kleijn [11] proved that the question is recursively solvable. In particular, they proved the following theorem:

Theorem 3. *$L(M)$ is palindromic if and only if $\{x \in L(M) \ : \ |x| < 3n\}$ is palindromic, where n is the number of states of M.*

For an NFA over an alphabet of at least 2 symbols, the $3n$ bound is easily seen to be optimal; for a DFA, however, the bound of $3n$ can be improved to $3n - 3$, and this is optimal.

While a naive implementation of Theorem 3 would take exponential time, in this section we show how to test palindromicity in polynomial time.

The main idea is to construct a "small" NFA M'_t, for some integer $t > 1$, where no word in $L(M'_t)$ is a palindrome, and M'_t accepts all non-palindromes of length $< t$ (in addition to some other non-palindromes). We omit the details of the construction (a similar construction appears in [25]).

Given an NFA M with n states, we now construct the cross-product with M'_{3n}, and obtain an NFA A that accepts $L(M) \ \cap \ L(M'_{3n})$. By Theorem 3, $L(A) = \emptyset$ if and only if $L(M)$ is palindromic. We can determine if $L(A) = \emptyset$ in linear time. If M has n states and t transitions, then A has $O(n^2)$ states and $O(tn)$ transitions. Hence we have proved the following theorem.

Theorem 4. *Let M be an NFA with n states and t transitions. The algorithm sketched above determines whether M accepts a palindromic language in $O(n^2 + tn)$ time.*

In analogy with Corollary 2 and using a different construction than that of Theorem 4, we also have the following proposition.

Proposition 2. *Given an NFA M with n states and t transitions, we can determine in $O(n^2 + t^2)$ time if M accepts infinitely many non-palindromes.*

5 Testing If an NFA Accepts a Word Matching a Pattern

In this section we consider the computational complexity of the decision problem:

NFA PATTERN ACCEPTANCE
INSTANCE: An NFA M over the alphabet Σ and a pattern p over some alphabet Δ.
QUESTION: Does there exist $x \in \Sigma^+$ such that $x \in L(M)$ and x matches p?

Since the pattern p is given as part of the input, this problem is actually somewhat more general than the sort of problem formulated as Question 1 of the introduction, where the language L was fixed.

The following result was proved by Restivo and Salemi [20] (a more detailed proof appears in [4]).

Theorem 5 (Restivo and Salemi). *Let L be a regular language and let Δ be an alphabet. The set P_Δ of all non-empty patterns $p \in \Delta^*$ such that p matches a word in L is effectively regular.*

Observe that Theorem 5 implies the decidability of the **NFA PATTERN ACCEPTANCE** problem. It is possible to give a boolean matrix based proof of Theorem 5 (see Zhang [27] for a study of this boolean matrix approach to automata theory) that provides an explicit description of an NFA accepting P_Δ, but due to space constraints we omit this proof. However, the reader may perhaps deduce the argument from the proof of the following algorithmic result, which uses similar ideas.

Theorem 6. *The* **NFA PATTERN ACCEPTANCE** *problem is PSPACE-complete.*

Proof (sketch). We first show that the problem is in PSPACE. By Savitch's theorem [23] it suffices to give an NPSPACE algorithm. Let $M = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{0, 1, \ldots, n-1\}$. For $a \in \Sigma$, let B_a be the $n \times n$ boolean matrix whose (i, j) entry is 1 if $j \in \delta(i, a)$ and 0 otherwise. Let $\mathcal{B}$ denote the semigroup generated by the B_a's. For $w = w_0 w_1 \cdots w_s \in \Sigma^*$, we write B_w to denote the matrix product $B_{w_0} B_{w_1} \cdots B_{w_s}$.

Let Δ be the set of letters occuring in p. We may suppose that $\Delta = \{1, 2, \ldots, k\}$. First, non-deterministically guess k boolean matrices $B_1, \ldots, B_k$. Next, for each i, verify that B_i is in the semigroup $\mathcal{B}$ by non-deterministically guessing a word w of length at most 2^{n^2} such that $B_i = B_w$. We guess w symbol-by-symbol and reuse space after perfoming each matrix multiplication while computing B_w. Then, if $p = p_0 p_1 \cdots p_r$, compute the matrix product $B = B_{p_0} B_{p_1} \cdots B_{p_r}$ and accept if and only if B describes an accepting computation of M.

To show hardness we reduce from the following PSPACE-complete problem [7, Problem AL6]. We leave the details to the reader.

DFA INTERSECTION
INSTANCE: An integer $k \geq 1$ and k DFAs $A_1, A_2, \ldots, A_k$, each over the alphabet Σ.
QUESTION: Does there exist $x \in \Sigma^*$ such that x is accepted by each A_i, $1 \leq i \leq k$? □

We may define various variations or special cases of the **NFA PATTERN ACCEPTANCE** problem, such as: **NFA ACCEPTS A k-POWER**, **NFA ACCEPTS INFINITELY MANY k-POWERS**, where each of these problems is defined in the obvious way. When k is part of the input (i.e., k is not fixed), these problems can be shown to be PSPACE-complete by a variation on the proof of Theorem 6. However, if k is fixed, both of these problems can be solved in polynomial time, as we now demonstrate.

Proposition 3. *Let M be an NFA with n states and t transitions, and set $N = n + t$, the size of M. For any fixed integer $k \geq 2$, there is an algorithm running in $O(n^{2k-1} t^k) = O(N^{2k-1})$ time to determine if M accepts a k-power.*

Proof (sketch). For a language $L \subseteq \Sigma^*$, we define $L^{1/k} = \{x \in \Sigma^* : x^k \in L\}$. It is well-known that if L is regular, then so is $L^{1/k}$. We leave it to the reader to verify that an NFA-ϵ M' accepting $L^{1/k}$ can be constructed with $n^{2k-1}+1$ states and at most t^k distinct transitions. Testing whether or not $L(M')$ accepts a non-empty word can be done in linear time, so the running time of our algorithm is $O(n^{2k-1} t^k)$. □

Corollary 4. *We can decide if an NFA M with n states and t transitions accepts infinitely many k-powers in $O(n^{2k-1} t^k)$ time.*

We may also consider the problems **NFA ACCEPTS A $\geq k$-POWER** and **NFA ACCEPTS INFINITELY MANY $\geq k$-POWERS**, again defined in the obvious way. Here, even for fixed k, these problems are both PSPACE-complete. Setting $k = 2$ corresponds to the problems **NFA ACCEPTS A POWER** and **NFA ACCEPTS INFINITELY MANY POWERS**, so we see that both these problems are PSPACE-complete as well.

To show PSPACE-hardness for the "infinitely many" problems, we reduce from the **DFA INTERSECTION INFINITENESS** problem, which is defined similarly to the **DFA INTERSECTION** problem, except that we now ask if there are infinitely many words x such that x is accepted by each A_i. This problem is easily seen to be PSPACE-complete as well.

6 Testing If an NFA Accepts a Non-k-Power

In the previous section we showed that it is computationally hard to test if an NFA accepts a k-power (when k is not fixed). In this section we show how to efficiently test if an NFA accepts a non-k-power. Again, we find it more congenial to discuss the opposite problem, which is whether an NFA accepts nothing but k-powers.

First, we need some classical results from combinatorics on words.

Theorem 7 (Lyndon and Schützenberger [15]). *If x, y, and z are words satisfying an equation $x^i y^j = z^k$, where $i, j, k \geq 2$, then they are all powers of a common word.*

Theorem 8 (Lyndon and Schützenberger [15]). *Let u and v be non-empty words. If $uv = vu$, then there exists a word x and integers $i, j \geq 1$, such that $u = x^i$ and $v = x^j$. In other words, u and v are powers of a common word.*

We include here the following combinatorial result, which, when applied to words in a regular language, gives a sort of "pumping lemma" for powers in a regular language.

Proposition 4. *Let u, v, and w be words, $v \neq \varepsilon$, and let $f, g \geq 1$ be integers, $f \neq g$. If $uv^f w$ and $uv^g w$ are non-primitive, then $uv^n w$ is non-primitive for all integers $n \geq 1$. Further, if uvw and $uv^2 w$ are k-powers for some integer $k \geq 2$, then v and $uv^n w$ are k-powers for all integers $n \geq 1$.*

The following result is an analogue of Theorem 3, from which we will derive an efficient algorithm for testing if a finite automaton accepts only k-powers.

Theorem 9. *Let L be accepted by an n-state NFA M and let $k \geq 2$ be an integer.*

1. *Every word in L is a k-power if and only if every word in the set $\{x \in L : |x| \leq 3n\}$ is a k-power.*
2. *All but finitely many words in L are k-powers if and only if every word in the set $\{x \in L : n \leq |x| \leq 3n\}$ is a k-power.*

Further, if M is a DFA over an alphabet of size ≥ 2, then the bound $3n$ may be replaced by $3n - 3$.

Ito et al. [12] proved a similar result for primitive words: namely, that if L is accepted by an n-state DFA over an alphabet of two or more letters and contains a primitive word, then it contains a primitive word of length $\leq 3n - 3$. In other words, every word in L is a power if and only if every word in the set $\{x \in L : |x| \leq 3n - 3\}$ is a power.

The proof of Theorem 9 is similar to that of [12, Proposition 7], albeit with some additional complications. We shall give a complete proof in the full version of this paper.

The characterization due to Ito et al. [12, Proposition 10] (see also Dömösi, Horváth, and Ito [6, Theorem 3]) of the regular languages consisting only of powers, along with Theorem 2, implies that any such language is slender. A simple application of the Myhill–Nerode Theorem gives the following weaker result.

Proposition 5. *Let L be a regular language and let $k \geq 2$ be an integer. If all but finitely many words of L are k-powers, then L is slender. In particular, if L is accepted by an n-state DFA and all words in L of length $\geq \ell$ are k-powers, then for all $r \geq \ell$, the number of words in L of length r is at most n.*

The following characterization is analogous to the characterization of palindromic regular languages given in [11, Theorem 8], and follows from Proposition 5, Theorem 2, and the (omitted) proof of Proposition 4.

Theorem 10. *Let $L \subseteq \Sigma^*$ be a regular language and let $k \geq 2$ be an integer. The language L consists only of k-powers if and only if it can be written as a finite union of languages of the form uv^*w, where $u, v, w \in \Sigma^*$ satisfy the following: there exists a primitive word $x \in \Sigma^*$ and integers $i, j \geq 0$ such that $v = x^{ik}$ and $wu = x^{jk}$.*

Next we apply Theorem 9 to deduce the following algorithmic result.

Theorem 11. *Let $k \geq 2$ be an integer. Given an NFA M with n states and t transitions, it is possible to determine if every word in $L(M)$ is a k-power in $O(n^3 + tn^2)$ time.*

Proof (sketch). We create an NFA, M'_r, for $r = 3n$, such that no word in $L(M'_r)$ is a k-power, and M'_r accepts all non-k-powers of length $\leq r$ (and perhaps some other non-k-powers).

Note that we may assume that $k \leq r$. If $k > r$, then no word of length $\leq r$ is a k-power. In this case, to obtain the desired answer it suffices to test if the set $\{x \in L(M) : |x| \leq r\}$ is empty. However, this set is empty if and only if $L(M)$ is empty, and this is easily verified in linear time.

We now form a new NFA A as the cross product of M'_r with M. From Theorem 9, it follows that $L(A) = \emptyset$ iff every word in $L(M)$ is a k-power. Again, we can determine if $L(A) = \emptyset$ in linear time.

We omit the details of the construction of M'_r, noting only that M'_r can be constructed to have at most $O(r^2)$ states and $O(r^2)$ transitions. After constructing the cross-product, this gives a $O(n^3 + tn^2)$ bound on the time required to determine if every word in $L(M)$ is a k-power. □

Theorem 9 suggests the following question: if M is an NFA with n states that accepts at least one non-k-power, how long can the shortest non-k-power be? Theorem 9 proves an upper bound of $3n$. A lower bound of $2n - 1$ for infinitely many n follows easily from the obvious $(n+1)$-state NFA accepting $\mathtt{a}^n(\mathtt{a}^{n+1})^*$, where n is divisible by k. However, Ito et al. [12] gave a very interesting example that improves this lower bound: if $x = ((ab)^n a)^2$ and $y = baxab$, then x and xyx

are squares, but $xyxyx$ is not a power. Hence, the obvious $(8n+8)$-state NFA that accepts $x(yx)^*$ has the property that the shortest non-k-power accepted is of length $20n+18$. We generalize this lower bound by defining x and y as follows: let $u = (ab)^n a$, $x = u^k$, and $y = x^{-1}(xbau^{-1}x)^k x^{-1}$. We leave it to the reader to deduce the following result.

Proposition 6. *Let $k \geq 2$ be fixed. There exist infinitely many NFAs M with the property that if M has r states, then the shortest non-k-power accepted is of length $(2 + \frac{1}{2k-2})r - O(1)$.*

We may also apply part (2) of Theorem 9 to obtain an algorithm to check if an NFA accepts infinitely many non-k-powers.

Theorem 12. *Let $k \geq 2$ be an integer. Given an NFA M with n states and t transitions, it is possible to determine if all but finitely many words in $L(M)$ are k-powers in $O(n^3 + tn^2)$ time.*

7 Automata Accepting Only Powers

We now move from the problem of testing if an automaton accepts only k-powers to that of testing if it accepts only powers (of any kind). Just as Theorem 9 was the starting point for our algorithmic results in Section 6, the following theorem of Ito et al. [12] (stated here in a slightly stronger form than in the original) is the starting point for our algorithmic results in this section.

Theorem 13. *Let L be accepted by an n-state NFA M.*

1. *Every word in L is a power if and only if every word in the set $\{x \in L : |x| \leq 3n\}$ is a power.*
2. *All but finitely many words in L are powers if and only if every word in the set $\{x \in L : n \leq |x| \leq 3n\}$ is a power.*

Further, if M is a DFA over an alphabet of size ≥ 2, then the bound $3n$ may be replaced by $3n-3$.

We next prove an analogue of Proposition 5. We need the following result, first proved by Birget [3], and later, independently, in a weaker form, by Glaister and Shallit [8].

Theorem 14. *Let $L \subseteq \Sigma^*$ be a regular language. Suppose there exists a set of pairs $S = \{(x_i, y_i) \in \Sigma^* \times \Sigma^* : 1 \leq i \leq n\}$ such that: (a) $x_i y_i \in L$ for $1 \leq i \leq n$; and (b) either $x_i y_j \notin L$ or $x_j y_i \notin L$ for $1 \leq i, j \leq n$, $i \neq j$. Then any NFA accepting L has at least n states.*

Proposition 7. *Let M be an n-state NFA and let ℓ be a non-negative integer such that every word in $L(M)$ of length $\geq \ell$ is a power. For all $r \geq \ell$, the number of words in $L(M)$ of length r is at most $7n$.*

Proof. We give the proof in full, as it illustrates an unusual and unexpected combination of techniques from both the theory of non-deterministic state complexity as well as the theory of combinatorics on words.

Let $r \geq \ell$ be an arbitrary integer. The proof consists of three steps.

Step 1. We consider the set A of words w in $L(M)$ such that $|w| = r$ and w is a k-power for some $k \geq 4$. For each such w, write $w = x^i$, where x is a primitive word, and define a pair (x^2, x^{i-2}). Let S_A denote the set of such pairs. Consider two pairs in S_A: (x^2, x^{i-2}) and (y^2, y^{j-2}). The word x^2y^{j-2} is primitive by Theorem 7 and hence is not in $L(M)$. The set S_A thus satifies the conditions of Theorem 14. Since $L(M)$ is accepted by an n-state NFA, we must have $|S_A| \leq n$ and thus $|A| \leq n$.

Step 2. Next we consider the set B of cubes of length r in $L(M)$. For each such cube $w = x^3$, we define a pair (x, x^2). Let S_B denote the set of such pairs. Consider two pairs in S_B: (x, x^2) and (y, y^2). Suppose that xy^2 and yx^2 are both in $L(M)$. The word xy^2 is certainly not a cube; we claim that it cannot be a square. Suppose it were. Then $|x|$ and $|y|$ are even, so we can write $x = x_1x_2$ and $y = y_1y_2$ where $|x_1| = |x_2| = |y_1| = |y_2|$. Now if $xy^2 = x_1x_2y_1y_2y_1y_2$ is a square, then $x_1x_2y_1 = y_2y_1y_2$, and so $y_1 = y_2$. Thus y is a square; write $y = z^2$. By Theorem 7, $yx^2 = z^2x^2$ is primitive, contradicting our assumption that $yx^2 \in L(M)$. It must be the case then that xy^2 is a k-power for some $k \geq 4$. Thus, $xy^2 = u^k$ for some primitive u uniquely determined by x and y. With each pair of cubes x^3 and y^3 such that both xy^2 and yx^2 are in $L(M)$ we may therefore associate a k-power $u^k \in L(M)$, where $k \geq 4$. We have already established in Step 1 that the number of such k-powers is at most n. It follows that by deleting at most n pairs from the set S_B we obtain a set of pairs satisfying the conditions of Theorem 14. We must therefore have $|S_B| \leq 2n$ and thus $|B| \leq 2n$.

Step 3. Finally we consider the set C of squares of length r in $L(M)$. For each such square $w = x^2$, we define a pair (x, x). Let S_C denote the set of such pairs. Consider two pairs in S_C: (x, x) and (y, y). Suppose that xy and yx are both in $L(M)$. The word xy is not a square and must therefore be a k-power for some $k \geq 3$. We write $xy = u^k$ for some primitive u uniquely determined by x and y. In Steps 1 and 2 we established that the number of k-powers of length r, $k \geq 3$, is $|A| + |B| \leq 3n$. It follows that by deleting at most $3n$ pairs from the set S_C we obtain a set of pairs satisfying the conditions of Theorem 14. We must therefore have $|S_C| \leq 4n$ and thus $|C| \leq 4n$.

Putting everything together, we see that there are $|A| + |B| + |C| \leq 7n$ words of length r in $L(M)$, as required. □

The bound of $7n$ in Proposition 7 is almost certainly not optimal. We now prove the following algorithmic result.

Theorem 15. *Given an NFA M with n states, it is possible to determine if every word in $L(M)$ is a power in $O(n^5)$ time.*

Proof (sketch). Checking if a word is a power can be done in linear time using the Knuth-Morris-Pratt algorithm [13]. By Theorem 13 and Proposition 7 it suffices to enumerate the words in $L(M)$ of lengths $1, 2, \ldots, 3n$, stopping if the number of such words in any length exceeds $7n$. If all these words are powers, then every word is a power. Otherwise, if we find a non-power, or if the number of words in any length exceeds $7n$, then not every word is a power. By the work of Mäkinen [16] or Ackerman & Shallit [1], we can enumerate these words in $O(n^5)$ time. $\square$

Using part (2) of Theorem 13 along with Proposition 7, one obtains the following in a similar manner.

Theorem 16. *Given an NFA M with n states, we can decide if all but finitely many words in $L(M)$ are non-powers in $O(n^5)$ time.*

8 Final Remarks

In this paper we examined the complexity of checking various properties of regular languages, such as consisting only of palindromes, containing at least one palindrome, consisting only of powers, or containing at least one power. In each case, we were able to provide an efficient algorithm or show that the problem is likely to be hard. Our results are summarized in the following table. We also report some upper and lower bounds on the length of a shortest palindrome, k-power, etc., accepted by an NFA; due to space constraints we must omit the proofs of these bounds. Here M is an NFA with n states and t transitions.

L	decide if $L(M) \cap L = \emptyset$	decide if $L(M) \cap L$ infinite	upper bound on shortest element of $L(M) \cap L$	worst-case lower bound known
palindromes	$O(n^2 + t^2)$	$O(n^2 + t^2)$	$2n^2 - 1$	$\frac{n^2}{2} - 3n + 5$
non-palindromes	$O(n^2 + tn)$	$O(n^2 + t^2)$	$3n - 1$	$3n - 1$
k-powers (k fixed)	$O(n^{2k-1}t^k)$	$O(n^{2k-1}t^k)$	kn^k	$\Omega(n^k)$
k-powers (k part of input)	PSPACE-complete	PSPACE-complete		
non-k-powers	$O(n^3 + tn^2)$	$O(n^3 + tn^2)$	$3n$	$(2 + \frac{1}{2k-2})n - O(1)$
powers	PSPACE-complete	PSPACE-complete	$(n+1)n^{n+1}$	$e^{\Omega(\sqrt{n \log n})}$
non-powers	$O(n^5)$	$O(n^5)$	$3n$	$\frac{5}{2}n - 2$

References

1. Ackerman, M., Shallit, J.: Efficient enumeration of regular languages. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 226–242. Springer, Heidelberg (2007)
2. Anderson, T., Rampersad, N., Santean, N., Shallit, J.: Finite automata, palindromes, patterns, and borders, `http://www.arxiv.org/abs/0711.3183`

3. Birget, J.-C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. 43, 185–190 (1992)
4. Castiglione, G., Restivo, A., Salemi, S.: Patterns in words and languages. Disc. Appl. Math. 144, 237–246 (2004)
5. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. 47, 149–158 (1986); Errata 302, 497–498 (2003)
6. Dömösi, P., Horváth, G., Ito, M.: A small hierarchy of languages consisting of non-primitive words. Publ. Math (Debrecen) 64, 261–267 (2004)
7. Garey, M., Johnson, D.: Computers and Intractability. Freeman, New York (1979)
8. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. 59, 75–77 (1996)
9. Heath-Brown, D.R.: Zero-free regions for Dirichlet L-functions, and the least prime in an arithmetic progression. Proc. Lond. Math. Soc. 64, 265–338 (1992)
10. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
11. Horváth, S., Karhumäki, J., Kleijn, J.: Results concerning palindromicity. J. Inf. Process. Cybern. EIK 23, 441–451 (1987)
12. Ito, M., Katsura, M., Shyr, H.J., Yu, S.S.: Automata accepting primitive words. Semigroup Forum 37, 45–52 (1988)
13. Knuth, D., Morris Jr., J., Pratt, V.: Fast pattern matching in strings. SIAM J. Computing 6, 323–350 (1977)
14. Kunze, M., Shyr, H.J., Thierrin, G.: h-bounded and semi-discrete languages. Information and Control 51, 147–187 (1981)
15. Lyndon, R.C., Schützenberger, M.-P.: The equation $a^m = b^n c^p$ in a free group. Michigan Math. J. 9, 289–298 (1962)
16. Mäkinen, E.: On lexicographic enumeration of regular and context-free languages. Acta Cybernetica 13, 55–61 (1997)
17. Martinez, A.: Efficient computation of regular expressions from unary NFAs. In: DCFS 2002, pp. 174–187 (2002)
18. Pritchard, P.: Linear prime-number sieves: a family tree. Sci. Comput. Programming 9, 17–35 (1987)
19. Păun, G., Salomaa, A.: Thin and slender languages. Disc. Appl. Math. 61, 257–270 (1995)
20. Restivo, A., Salemi, S.: Words and patterns. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 215–218. Springer, Heidelberg (2002)
21. Rosaz, L.: Puzzle corner, #50. Bull. European Assoc. Theor. Comput. Sci. 76, 234 (February 2002); Solution 77, 261 (June 2002)
22. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages. Springer, Berlin (1997)
23. Savitch, W.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. System Sci. 4, 177–192 (1970)
24. Shallit, J.: Numeration systems, linear recurrences, and regular sets. Inform. Comput. 113, 331–347 (1994)
25. Shallit, J., Breitbart, Y.: Automaticity I: Properties of a measure of descriptional complexity. J. Comput. System Sci. 53, 10–25 (1996)
26. Yu, S.: Regular languages. In: Handbook of Formal Languages, Ch. 2, pp. 41–110 (1997)
27. Zhang, G.-Q.: Automata, Boolean matrices, and ultimate periodicity. Inform. Comput. 152, 138–154 (1999)