Undecidability and Finite Automata

Jörg Endrullis[1], Jeffrey Shallit[2(✉)], and Tim Smith[2]

[1] Department of Computer Science, Vrije Universiteit Amsterdam,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
joerg@endrullis.de
[2] School of Computer Science, University of Waterloo,
Waterloo, ON N2L 3G1, Canada
{shallit,timsmith}@uwaterloo.ca

Abstract. Using a novel rewriting problem, we show that several natural decision problems about finite automata are undecidable (i.e., recursively unsolvable). In contrast, we also prove three related problems are decidable. We apply one result to prove the undecidability of a related problem about k-automatic sets of rational numbers.

Keywords: Finite automata · Undecidability · Conjugate · Power

1 Introduction

Starting with the first result of Turing [16], computer scientists have assembled a large collection of natural decision problems that are undecidable (i.e., recursively unsolvable); see, for example, the book of Rozenberg and Salomaa [12].

Although some of these results deal with relatively weak computing models, such as pushdown automata [2,6], few, if any, are concerned with the very simplest model: the finite automaton. One exception is the following decision problem, due to Engelfriet and Rozenberg [5, Theorem 15]: given a finite automaton M with an input alphabet of both primed and unprimed letters (i.e., an alphabet $\Sigma \cup \Sigma'$, where $\Sigma' = \{a' : a \in \Sigma\}$), decide if M accepts a word w where the primed letters, after the primes have been removed, form a word identical to that formed by the unprimed letters. This problem was also mentioned by Hoogeboom [7]. This problem is easily seen to be undecidable, as it is a disguised version of the classical Post correspondence problem [10].

In this paper we start by proving a novel lemma on rewriting systems. In Sect. 3, this lemma is then applied to give a new example of a natural problem on finite automata that is undecidable. In Sect. 4 we prove that a related problem on the so-called k-automatic sets of rational numbers is also undecidable. In Sect. 5 we prove the undecidability of yet another problem about finite automata. Finally, in Sect. 5 we show that it is decidable if a finite automaton accepts two distinct conjugates.

É. Charlier et al. (Eds.): DLT 2017, LNCS 10396, pp. 160–172, 2017.
DOI: 10.1007/978-3-319-62809-7_11

2 A Lemma on Rewriting Systems

For our purposes, a *rewriting system* S over an alphabet Σ consists of a finite set of context-free rules of the form $\ell \to r$, where $\ell, r \in \Sigma^*$. Such a rewriting rule applies to the word $\alpha\ell\beta \in \Sigma^*$, and converts it to $\alpha r\beta$. We indicate this by writing $\alpha\ell\beta \Longrightarrow \alpha r\beta$. We use $\Longrightarrow^*$ for the reflexive, transitive closure of $\Longrightarrow$ (so that $\gamma \Longrightarrow^* \zeta$ means that there is a sequence of 0 or more rules taking the word γ to ζ). A rewriting system is said to be *length-preserving* if $|\ell| = |r|$ for all rewriting rules $\ell \to r$.

Many undecidable decision problems related to rewriting systems are known [3]. However, to the best of our knowledge, the following one is new.

REWRITE-POWER

Instance: An alphabet Σ containing the symbols a and b (and possibly other symbols), and a length-preserving rewriting system S.

Question: Does there exist an integer $n \geq 1$ such that $a^n \Longrightarrow^* b^n$?

Lemma 1. *The decision problem* REWRITE-POWER *is undecidable.*

Proof. The standard approach for showing that a rewriting problem is undecidable is to reduce from the halting problem, by encoding a Turing machine M and simulating its computation using the rewriting rules; for example, see [3].

The difficulty with applying that approach in the present case is the lack of asymmetry, i.e., the fact that the initial word consists of all a's. Because there is no distinguished symbol with which to start the simulation, unwanted parallel simulations of M could occur at different parts of the word.

To deal with this difficulty, we construct a rewriting system that permits multiple simulations of M to arise, but employs a delimiter symbol \$ to ensure that they do not interfere with each other. Each simulation works on its own portion of the word, and changes it to b's (ending by changing the delimiter symbol as well) only if M halts.

Here are the details. We use the one-tape model of Turing machine from Hopcroft and Ullman [8], where $M = (Q, \Omega, \Gamma, \delta, q_0, \mathtt{B}, q_f)$. Here Q is the set of states of M, with $q_0 \in Q$ the start state and $q_f \in Q$ the unique final state. Let Ω be the input alphabet and Γ the tape alphabet, with $\mathtt{B} \in \Gamma$ being the distinguished blank symbol. Let δ be the (partial) transition function, with domain $Q \times \Gamma$ and range $Q \times \Gamma \times \{L, R\}$. We assume without loss of generality that M halts, i.e., M has no next move, iff it is in state q_f. We also assume that $a, b, \$ \notin \Gamma$.

We construct our length-preserving rewriting system mimicking the computations of M as follows. Let $\Sigma = \Gamma \cup Q \cup \{a, b, \$\}$. Let S contain the following rewriting rules:

$$aa \to \$q_0 \tag{1}$$
$$a \to \mathtt{B} \tag{2}$$
$$q_i c \to d q_j \quad \text{if } (q_j, d, R) \in \delta(q_i, c) \text{ with } c, d \in \Gamma \tag{3}$$
$$f q_i c \to q_j f d \quad \text{if } (q_j, d, L) \in \delta(q_i, c) \text{ and } f \in \Gamma \tag{4}$$
$$q_f c \to c q_f \quad \text{for all } c \in \Gamma \tag{5}$$
$$c q_f \to q_f b \quad \text{for all } c \in \Gamma \tag{6}$$
$$\$q_f \to bb \tag{7}$$

Rule (1) starts a new simulation. Each simulation has its own head symbol q_i and left end-marker \$, and never moves the head symbol past an a, b, or \$. This ensures that the simulations are kept separate from each other. Rule (2) converts an a to a blank symbol, available for use in a simulation. Rules (3) and (4) are used to simulate the transitions of M. Once a simulation reaches q_f (meaning that M has halted), its head can be moved to the right using rule (5), past all of the symbols it has read, and then back to the left using rule (6), changing all of those symbols to b's. Finally, the simulation can be stopped using rule (7).

We argue that M halts when run on a blank tape iff $a^n \Longrightarrow^* b^n$ for some $n \geq 1$.

If M halts when run on a blank tape, then there is some number of tape cells k which it uses. Let $n = k+2$. Then S can use rule (1) to start a simulation with the two a's at the left end of the word, use rule (2) to convert the k remaining a's to blank cells, and run the simulation using rules (3) and (4). Eventually M halts in the final state q_f. At that point, S can move the head to the right end of the word using rule (5), convert all k tape cells to b's using rule (6), and then convert the end-marker \$ and tape head to b's with rule (7). Thus we have $a^n \Longrightarrow^* b^n$.

For the other direction, if the initial word a^n is ever transformed into b^n, it means that one or more simulations were run, each of which operated on a portion of the word without interference from the others. The absence of interference can be deduced from the shape of the rules. There is only one occurrence of the marker \$ in the left-hand sides of the rules, namely as the leftmost symbol in the left-hand side of rule (7). Furthermore, \$ can only be rewritten to b which does not occur in any left-hand side.

Each simulation runs M on a blank tape, and uses a number of tape cells bounded by the length of its portion of the word. Since every portion of the word was transformed into b's, M halted in every one of the simulations (otherwise the word would still contain one or more head symbols and end-markers). The completion of any one of these simulations is enough to show that M halts when run on a blank tape.

Therefore M halts when run on a blank tape iff there exists an $n \geq 1$ such that $a^n \Longrightarrow^* b^n$, completing the reduction from the halting problem. Since the halting problem is undecidable, `REWRITE-POWER` is also undecidable. □

3 An Undecidable Problem on Finite Automata

Our model of finite automaton is the usual one (e.g., [8]). We now consider a decision problem on finite automata. To state it, we need the notion of the product of two words of the same length. Let Σ, Δ be alphabets, and let $w \in \Sigma^*$, $x \in \Delta^*$, with $|w| = |x|$. Then by $w \times x$ we mean the word over the alphabet $\Sigma \times \Delta$ whose projection π_1 over the first coordinate is w and whose projection π_2 over the second coordinate is x. More precisely, if $w = a_1 a_2 \cdots a_n$ and $x = b_1 b_2 \cdots b_n$, then $w \times x = [a_1, b_1][a_2, b_2] \cdots [a_n, b_n]$. In this case $\pi_1(w \times x) = w$ and $\pi_2(w \times x) = x$. For example, if $y = [\mathtt{t}, \mathtt{h}]\,[\mathtt{e}, \mathtt{o}]\,[\mathtt{r}, \mathtt{e}]\,[\mathtt{m}, \mathtt{s}]$, then $\pi_1(y) = \mathtt{term}$ and $\pi_1(y) = \mathtt{hoes}$. To simplify notation, we often write $\binom{w}{x}$ in place of $w \times x$. For example,

$$\begin{bmatrix} \mathtt{cat} \\ \mathtt{dog} \end{bmatrix} \quad \text{means the same thing as} \quad \begin{bmatrix} \mathtt{c} \\ \mathtt{d} \end{bmatrix} \begin{bmatrix} \mathtt{a} \\ \mathtt{o} \end{bmatrix} \begin{bmatrix} \mathtt{t} \\ \mathtt{g} \end{bmatrix} \quad \text{and} \quad [\mathtt{c}, \mathtt{d}][\mathtt{a}, \mathtt{o}][\mathtt{t}, \mathtt{g}].$$

Consider the following decision problem:

ACCEPTS-SHIFT

Instance: An alphabet Γ, a letter $c \notin \Gamma$, and a finite automaton M with input alphabet $(\Gamma \cup \{c\})^2$.

Question: Does M accept a word of the form $xc^n \times c^n x$ for some $x \in \Gamma^*$ and $n \geq 0$?

Theorem 2. *The decision problem* ACCEPTS-SHIFT *is undecidable.*

Proof. We reduce from the problem REWRITE-POWER. An instance of this decision problem is a set S of length-preserving rewriting rules, an alphabet Σ, and letters $a, b \in \Sigma$. Define $\Gamma = \Sigma \cup \{d\}$, where $d \notin \Sigma$ is a new symbol. Now we define the following regular languages:

$$E = \left\{ \begin{bmatrix} e \\ e \end{bmatrix} : e \in \Sigma \right\} \quad \text{and} \quad R = \left\{ \begin{bmatrix} r \\ \ell \end{bmatrix} : \ell \to r \in S \right\}$$

$$L = \begin{bmatrix} d \\ c \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}^+ \begin{bmatrix} d \\ d \end{bmatrix} \left(E^* R E^* \begin{bmatrix} d \\ d \end{bmatrix} \right)^* \begin{bmatrix} c \\ b \end{bmatrix}^+ \begin{bmatrix} c \\ d \end{bmatrix}.$$

Let $M = (Q, \Delta, \delta, q_0, F)$ be a deterministic finite automaton accepting L, with $\Delta = (\Gamma \cup \{c\})^2$. Clearly M can be constructed effectively from the definitions.

We claim that, for all $n \geq 2$, we have $a^{n-1} \Longrightarrow^* b^{n-1}$ iff the language $L = L(M)$ contains a word of the form $xc^n \times c^n x$. The crucial observation is that

$$u \Longrightarrow v \quad \text{iff} \quad \begin{bmatrix} v \\ u \end{bmatrix} \in E^* R E^*. \tag{8}$$

This follows immediately from the definitions of E and R.

$\Longrightarrow$: Suppose

$$u_0 := a^{n-1} \Longrightarrow u_1 \Longrightarrow \cdots \Longrightarrow u_m = b^{n-1}$$

with $m \geq 1$ and $n \geq 2$. Then

$$\begin{bmatrix} u_{i+1} \\ u_i \end{bmatrix} \in E^* R E^*$$

for $0 \leq i < m$. Then

$$\begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_1 \\ u_0 \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_2 \\ u_1 \end{bmatrix} \cdots \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_m \\ u_{m-1} \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \in \begin{bmatrix} d \\ d \end{bmatrix} \left(E^* R E^* \begin{bmatrix} d \\ d \end{bmatrix} \right)^*.$$

Hence

$$\begin{bmatrix} d \\ c \end{bmatrix} \begin{bmatrix} u_0 \\ c^{n-1} \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_1 \\ u_0 \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_2 \\ u_1 \end{bmatrix} \cdots \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} u_m \\ u_{m-1} \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} c^{n-1} \\ u_m \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} \in L,$$

as desired. The first component is $du_0du_1d \cdots du_mdc^n$, while the second component is $c^n du_0du_1 \cdots du_{m-1}du_md$. Taking $x = du_0du_1d \cdots du_md$, we see that $xc^n \times c^n x \in L$.

$\Longleftarrow$: Assume that $xc^n \times c^n x \in L$ for some word x with $n \geq 2$. Now L consists only of words of the form

$$w = \begin{bmatrix} d \\ c \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix}^i \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} v_0 \\ u_0 \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} v_1 \\ u_1 \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \cdots \begin{bmatrix} v_m \\ u_m \end{bmatrix} \begin{bmatrix} d \\ d \end{bmatrix} \begin{bmatrix} c \\ b \end{bmatrix}^j \begin{bmatrix} c \\ d \end{bmatrix}$$

where $u_t \Longrightarrow v_t$ for $1 \leq t \leq m$ and $i, j \geq 1$. Observe that

$$\pi_1(w) = da^i dv_0 dv_1 \cdots dv_m dc^{j+1} \quad \text{and} \quad \pi_2(w) = c^{i+1} du_0 du_1 \cdots du_m db^j d,$$

so if $\pi_1(w) = xc^n$ and $\pi_2(w) = c^n x$ we must have $i = j = n - 1$ and $x = da^i dv_0 dv_1 \cdots dv_m d = du_0 du_1 \cdots du_m db^j d$. Since d is a new symbol, not in the alphabet of Σ, it follows that $u_0 = a^i$, $u_1 = v_0$, $u_2 = v_1, \ldots$, $u_m = v_{m-1}$, and $b^j = v_m$.

But then $u_0 \Longrightarrow v_0 = u_1$, $u_1 \Longrightarrow v_1 = u_2$, and so forth, up to $u_{m-1} \Longrightarrow v_{m-1} = u_m$, and finally $u_m \Longrightarrow v_m = b^j$. So $u_0 \Longrightarrow^* v_m$, and therefore $a^{n-1} \Longrightarrow^* b^{n-1}$. This completes the proof. □

Remark 3. In the decision problem `ACCEPTS-SHIFT`, the undecidability of the problem arises, in an essential way, from words of the form $xc^n \times c^n x$ where $n < |x|$, and not from those words with $n \geq |x|$ as one might first suspect. More formally, the related decision problem defined below is actually solvable in cubic time.

`ACCEPTS-LONG-SHIFT`

Instance: An alphabet Σ, a letter $c \notin \Sigma$, and a finite automaton M with input alphabet $(\Sigma \cup \{c\})^2$.

Question: Does M accept a word of the form $xc^n \times c^n x$ for some $n \geq |x|$?

Theorem 4. *The decision problem* ACCEPTS-LONG-SHIFT *is solvable in cubic time.*

Proof. Suppose $x = a_1 a_2 \cdots a_m$. If $y = xc^n \times c^n x$ and $n \geq |x|$ then

$$y = [a_1, c] \cdots [a_m, c][c, c]^{n-m}[c, a_1][c, a_2] \cdots [c, a_m].$$

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, we can create a nondeterministic finite automaton M' that accepts all x for which the corresponding y is accepted by M. The idea is that M' has state set $Q' = Q \times Q \times Q$; on input x the machine M' "guesses" a state $q \in Q$, and stores it in the second component, and then simulates M on input $x \times c^m$ in the first component, starting from q_0 and reaching some state p, and simulates M on input $c^m \times x$ in the third component, starting from q. Finally, M' accepts if the third component is an element of F and if there exists a path from p to q labeled $[c, c]^i$ for some $i \geq 0$. Now we can test whether M' accepts a word by using depth-first or breadth-first search on the transition diagram of M', whose size is at most cubic in terms of the size of M. □

4 Application to *k*-automatic Sets of Rational Numbers

Recently the second author and co-authors defined a notation of k-automaticity for sets of non-negative rational numbers [11,13], in analogy with the more well-known concept for sets of non-negative integers [4].

For an integer $k \geq 2$ define $\Sigma_k = \{0, 1, \ldots, k-1\}$. If $w \in \Sigma_k^*$, define $[w]_k$ to be the integer represented by the word w in base k (assuming the most significant digit is at the left). Let M be a finite automaton with input alphabet $\Sigma_k \times \Sigma_k$. We define $\mathrm{quo}_k(M) \subseteq \mathbb{Q}^{\geq 0}$ to be the set

$$\left\{ \frac{[\pi_1(x)]_k}{[\pi_2(x)]_k} : x \in L(M) \right\}.$$

Furthermore, we call a set $T \subseteq \mathbb{Q}^{\geq 0}$ k-automatic if there exists a finite automaton M such that $T = \mathrm{quo}_k(M)$.

We first consider the following decision problem:

ACCEPTS-POWER

Instance: An integer $k \geq 2$, and a finite automaton M with input alphabet $(\Sigma_k)^2$.

Question: Is $\mathrm{quo}_k(L(M)) \cap \{k^i : i \geq 0\}$ nonempty?

Theorem 5. *The problem* ACCEPTS-POWER *is undecidable.*

Proof. The basic idea is to reduce once more from REWRITE-POWER, using the same construction as in the proof of Theorem 2. Our reduction produces an instance of ACCEPTS-SHIFT consisting of an alphabet Γ of cardinality ℓ, a letter

$c \notin \Gamma$, and a finite automaton M. By renaming symbols, if necessary, we can assume the symbols of Γ are the digits $1, 2, \ldots, \ell$ and c is the digit 0. It then suffices to take $k = \ell + 1$. Then $y \in L(M)$ with $\mathrm{quo}_k(y)$ a power of k if and only if $y = x0^n \times 0^n x$ for some x and some $n \geq 0$. Note that, by our construction in the proof of Theorem 2, if M accepts $x0^n \times 0^n x$, then x contains no 0's. □

Now consider a family of analogous decision problems ACCEPTS-POWER(k), where in each problem k is fixed.

Theorem 6. *For each integer $k \geq 2$, the decision problem* ACCEPTS-POWER(k) *is undecidable.*

Proof. We have to overcome the problem that k can depend on the size of Γ. To do so, we recode all words over the alphabet $\{0, 1\}$. It suffices to use the morphism φ defined by

$$\varphi(c) = 0^{m+1} \qquad\qquad \varphi(a_i) = 1^i 0^{m-i} 1$$

where $\Sigma = \{a_1, a_2, \ldots, a_m\}$. In the proof of Theorem 2, we replace E, R, L by E', R', L', as follows:

$$E' = \left\{ \begin{bmatrix} \varphi(e) \\ \varphi(e) \end{bmatrix} : e \in \Sigma \right\} \quad \text{and} \quad R' = \left\{ \begin{bmatrix} \varphi(r) \\ \varphi(\ell) \end{bmatrix} : \ell \to r \in S \right\}$$

$$L' = \begin{bmatrix} \varphi(d) \\ \varphi(c) \end{bmatrix} \begin{bmatrix} \varphi(a) \\ \varphi(c) \end{bmatrix}^+ \begin{bmatrix} \varphi(d) \\ \varphi(d) \end{bmatrix} \left(E^* R E^* \begin{bmatrix} \varphi(d) \\ \varphi(d) \end{bmatrix} \right)^* \begin{bmatrix} \varphi(c) \\ \varphi(b) \end{bmatrix}^+ \begin{bmatrix} \varphi(c) \\ \varphi(d) \end{bmatrix}.$$

The construction works because the blocks for symbols of Σ begin and end with at least one 1, while the block for c consists of all 0's. Therefore, if the first coordinate of an element of L' has a suffix in 0^+, this can only arise from $\varphi(c)$, and the same for prefixes of the second coordinate. □

5 Problems About Conjugates

Recall that we say two words x and y are *conjugates* if one is a cyclic shift of the other; that is, if there exist u, v such that $x = uv$ and $y = vu$.

The undecidability result of the previous section suggests studying the following related natural decision problem.

ACCEPTS-GENERAL-SHIFT

Instance: A finite automaton M with input alphabet Σ^2.

Question: Does M accept a word of the form $x \times y$ for conjugates $x, y \in \Sigma^*$?

Theorem 7. *The decision problem* ACCEPTS-GENERAL-SHIFT *is undecidable.*

Proof. We reduce from the problem ACCEPTS-SHIFT. An instance of this problem is an alphabet Γ, a letter $c \notin \Gamma$, and a finite automaton M with input alphabet $(\Gamma \cup \{c\})^2$.

First check whether M accepts a word of the form $x \times x$ for some $x \in \Gamma^*$. (This is decidable because the language $\{x \times x : x \in \Gamma^*\}$ is regular.) If so, ACCEPTS-SHIFT(Γ, c, M) = "yes". Otherwise, construct a finite automaton M' whose language is

$$L(M) \cap \{sc^+ \times c^+t \mid s, t \in \Gamma^*\}.$$

Notice that ACCEPTS-SHIFT(Γ, c, M') = ACCEPTS-SHIFT(Γ, c, M). Clearly we have that if ACCEPTS-GENERAL-SHIFT(M') = "no", then ACCEPTS-SHIFT (Γ, c, M') = "no".

So suppose that ACCEPTS-GENERAL-SHIFT(M') = "yes". Then M' accepts a word $w = x \times y$ for words $x = uv$, $y = vu$ where $u, v \in (\Gamma \cup \{c\})^*$. We now show that $w = zc^n \times c^n z$ for some $z \in \Gamma^*$ and $n \geq 1$.

By the construction of M', uv ends with c, vu begins with c, and any two occurrences of c in uv or vu have only c's between them. Hence if u or v is empty, then $w = c^n \times c^n$ for some $n \geq 1$, and we can take $z = \epsilon$, the empty word. So say neither u nor v is empty. Then v begins and ends with c, and hence v is in c^+. It follows that if u contains c, then u begins and ends with c, so again $w = c^n \times c^n$ for some $n \geq 1$, and we can take $z = \epsilon$. So say u does not contain c. Then $w = uc^n \times c^n u$ with $u \in \Gamma^+$ and $n = |v|$, and we can take $z = u$.

So $w = zc^n \times c^n z$ for some word $z \in \Gamma^*$ and $n \geq 1$. Therefore we have ACCEPTS-SHIFT(Γ, c, M') = "yes". This completes the reduction. Then since ACCEPTS-SHIFT is undecidable by Theorem 2, ACCEPTS-GENERAL-SHIFT is also undecidable. □

Now we turn to two other decision problems, both inspired by the problem ACCEPTS-GENERAL-SHIFT. The first is

ACCEPTS-DISTINCT-CONJUGATES

Instance: A DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Question: Does M accept two distinct conjugates uv and vu?

We will prove

Theorem 8. ACCEPTS-DISTINCT-CONJUGATES *is decidable.*

To prove this theorem, we need the concept of primitive word and primitive root. A nonempty word x is said to be *primitive* if it cannot be written in the form $x = y^i$ for a word y and an integer $i \geq 2$. The *primitive root* of a word x is the unique primitive word t such that $x = t^j$ for some $j \geq 1$.

Lemma 9. *If a DFA M of n states accepts two distinct conjugates, then it accepts two distinct conjugates uv and vu, with at least one of u and v of length $\leq n^2$.*

Proof. Let $L = L(M)$, the language accepted by $M = (Q, \Sigma, \delta, q_0, F)$, where $|Q| = n$. Suppose that there exist $uv \in L$, $vu \in L$, but $uv \neq uv$. Without loss of generality, assume $|uv|$ is as small as possible. Assume, contrary to what we want to prove, that both $|u|$ and $|v|$ are $> n^2$.

Consider the acceptance path of uv through M: it looks like $\delta(q_0, u) = q_1$ and $\delta(q_1, v) = p_1$ for some $q_1 \in Q$ and $p_1 \in F$. Similarly, consider the acceptance path of vu through M: it looks like $\delta(q_0, v) = q_2$ and $\delta(q_2, u) = p_2$ for some $q_2 \in Q$ and $p_2 \in F$.

Now create a new DFA $M' = (Q \times Q, \Sigma, \delta', q_0', F')$ by the usual product construction, where $\delta'([r, s], a) := [\delta(r, a), \delta(s, a)]$ and $q_0' = [q_0, q_1]$ and $F = \{[q_2, p_1]\}$. Then M' has n^2 states and accepts v.

Since $|v| > n^2$, the acceptance path for v in M' visits $\geq n^2 + 2$ states and hence some state is repeated, giving us a loop of at most n^2 states that can be cut out. Hence we can write $v = v_1 v_2 v_3$, where $v_2 \neq \epsilon$ and $v_1 v_3 \neq \epsilon$, and M' accepts $v_1 v_3$. In M, then, it follows that $\delta(q_1, v_1 v_3) = p_1$ and $\delta(q_0, v_1 v_3) = q_2$, and hence M accepts the conjugates $uv_1 v_3$ and $v_1 v_3 u$. Since $|uv_1 v_3| < |uv|$, the minimality of $|uv|$ implies that these conjugates cannot be distinct, and so we must have

$$uv_1 v_3 = v_1 v_3 u. \tag{9}$$

We can now repeat the argument of the previous paragraph for the word u. We get a decomposition $u = u_1 u_2 u_3$ where $u_2 \neq \epsilon$ and $u_1 u_3 \neq \epsilon$, and we get

$$vu_1 u_3 = u_1 u_3 v. \tag{10}$$

Finally, the acceptance paths in M we have created imply that we can cut out both u_2 and v_2 simultaneously from uv and vu, and still get words accepted by M. So $u_1 u_3 v_1 v_3$ and $v_1 v_3 u_1 u_3$ are both accepted. Again, by minimality, we get that

$$u_1 u_3 v_1 v_3 = v_1 v_3 u_1 u_3. \tag{11}$$

Now, by the Lyndon-Schützenberger theorem (see, e.g., [9,14]), Eq. (11) implies the existence of a nonempty word t and integers i, j such that $u_1 u_3 = t^i$, $v_1 v_3 = t^j$. Without loss of generality, we can assume that t is primitive.

Applying the same theorem to Eq. (10) tells us that there exists k such that $v = t^k$. And applying the same theorem once more to Eq. (9) tells us that there exists ℓ such that $u = t^\ell$. But then $uv = vu$, a contradiction. □

Remark 10. We observe that the bound of n^2 in the previous result is optimal, up to a constant multiplicative factor. Consider the languages

$$L_t = (a^t)^+ b (a^{t+1})^+ bb \cup (a^t)^+ bb (a^{t+1})^+ bb.$$

Then it is easy to see that L_t can be accepted by a (complete) DFA of $n = 3t + 8$ states. The shortest pair of distinct conjugates in L_n, however, are $a^{t(t+1)} b a^{t(t+1)} bb$ and $a^{t(t+1)} bb a^{t(t+1)} b$, corresponding to $u = a^{t(t+1)} b$ of length $t^2 + t + 1$ and $v = a^{t(t+1)} bb$ of length $t^2 + t + 2$. Thus both u and v are of length $n^2/9 + O(n)$.

We can now prove Theorem 8.

Proof. Given $L = L(M)$, for each nonempty word x define the language

$$L_x = \{y \in \Sigma^* : xy \in L,\ yx \in L,\ xy \neq yx\}.$$

We observe that each L_x is a regular language. To see this, note that we can write $L_x = L_1 \cap L_2 \cap L_3$, where

$$L_1 = \{y \in \Sigma^* : xy \in L\}$$
$$L_2 = \{y \in \Sigma^* : yx \in L\}$$
$$L_3 = \{y \in \Sigma^* : xy \neq yx\}.$$

Both L_1 and L_2 are easily seen to be regular, and finite automata accepting them are easily constructed from M. To see that the same holds for L_3, note that if $xy = yx$ with x nonempty, then by the Lyndon-Schützenberger theorem it follows that $y \in t^*$, where t is the primitive root of x. Hence $L_3 = \overline{t^*}$. Therefore we can construct a finite automaton M_x accepting L_x.

Finally, here is the decision procedure. By Lemma 9 we know that if an n-state DFA M accepts a pair of words uv and vu with $uv \neq vu$, then it must accept a pair with either $|u| \leq n^2$ or $|v| \leq n^2$. Thus, it suffices to enumerate all $u \in \Sigma^*$ of lengths $1, 2, \ldots, n^2$, and compute M_u for each u. If at least one M_u has $L(M_u)$ nonempty, then answer "yes"; otherwise answer "no". □

An alternative approach for proving Theorem 8 was suggested by the referee, as follows:

Proof. We show that `ACCEPTS-DISTINCT-CONJUGATES` is decidable by reducing it to the functionality problem for nondeterministic streaming string transducers (NSSTs) [1]. An NSST is a one-way nondeterministic automaton equipped with a fixed set of variables in which to store strings. At each step, it reads a symbol from the input, changes state, and updates its string variables in parallel with a "copyless assignment". At the end of the input, it produces an output string based on its string variables and final state. See [1] for details.

Consider the relation R defined as the set of pairs $\{(uv, vu) \in L(M) \times L(M)\}$. An NSST T can implement R as a transduction as follows. T uses two string variables X and Y. Let w be T's input. T updates X with $X := X\sigma$ whenever a symbol σ of w is read, until some nondeterministic transition, after which, as long as symbols σ of w are read, the following updates are performed: $X := X$ and $Y := Y\sigma$. When the end of w is reached, $w = uv$ for $u = X$ and $v = Y$. During the computation, T uses its finite-state control to check that w is in $L(M)$. At the same time, T checks that vu is in $L(M)$ by guessing a state q of M, simulating M on u starting from q, verifying that M ends u in an accepting state, simulating M on v starting from q_0, and verifying that M ends v in q. If all of these checks succeed, T outputs $YX = vu$.

We say that T is *functional* if for every input string, T produces at most one output string. If T is functional, then for every $x \in L(M)$, x has at most

one conjugate in $L(M)$. Then since every string is a conjugate of itself, x has no conjugates other than x in $L(M)$, and so M does not accept distinct conjugates. On the other hand, if M does not accept distinct conjugates, then for each input x, T can only produce x as output, and so T is functional. Therefore T is functional iff the answer to `ACCEPTS-DISTINCT-CONJUGATES` is "no". By Theorem 4.1 of [1], checking if T is functional is decidable. Therefore `ACCEPTS-DISTINCT-CONJUGATES` is decidable. □

Our second decision problem is

`ACCEPTS-NON-CONJUGATES`

Instance: A DFA $M = (Q, \Sigma, \delta, q_0, F)$.

Question: Does M accept two words of the same length that are not conjugates?

We prove

Theorem 11. `ACCEPTS-NON-CONJUGATES` *is decidable.*

Proof. Given a formal language L over an ordered alphabet Σ, we define $\text{lexlt}(L)$ to be the union, over all $n \geq 0$, of the lexicographically least word of length n in L, if it exists. As is well-known (see, e.g., [15, Lemma 1]), if L is regular, then so is $\text{lexlt}(L)$. Furthermore, given a DFA for L, we can algorithmically construct a DFA for $\text{lexlt}(L)$.

We also define $\text{cyc}(L)$ to be the union, over all words $w \in L$, of the conjugates of w. Again, as is well-known (see, e.g., [14, Theorem 3.4.3]), if L is regular, then so is $\text{cyc}(L)$. Furthermore, given a DFA for L, we can algorithmically construct a DFA for $\text{cyc}(L)$.

We claim that L contains two words x and y of the same length that are non-conjugates if and only if L is not a subset of $\text{cyc}(\text{lexlt}(L))$.

Suppose such x, y exist. Let t be the lexicographically least word in L of length $|x|$. If t is a conjugate of x, then y is not a conjugate of t, so $y \notin \text{cyc}(\text{lexlt}(L))$. On the other hand, if t is not a conjugate of x, then $x \notin \text{cyc}(\text{lexlt}(L))$. In both cases L is not a subset of $\text{cyc}(\text{lexlt}(L))$.

Suppose L is not a subset of $\text{cyc}(\text{lexlt}(L))$. Then there is some word of some length n in L, say x, that is not a conjugate of the lexicographically least word of length n, say y. Then x and y are the desired two words.

Putting this all together, we get our decision procedure for the decision problem `ACCEPTS-NON-CONJUGATES`: given the DFA M for L, construct the DFA M' for $L - \text{cyc}(\text{lexlt}(L))$ using the techniques mentioned above. If M' accepts at least one word, then the answer for `ACCEPTS-NON-CONJUGATES` is "yes"; otherwise it is "no". □

6 Final Remarks

We still do not know whether the following problem from [11, p. 363] is decidable:

ACCEPTS-INTEGER

Instance: A finite automaton M with input alphabet $(\Sigma_k)^2$.

Question: Is $\mathrm{quo}_k(L(M)) \cap \mathbb{N}$ nonempty?

Unfortunately our techniques do not seem immediately applicable to this problem.

We mention two other problems about finite automata whose decidability is still open:

1. Given a DFA M with input alphabet $\{0, 1\}$, decide if there exists at least one prime number p such that M accepts the base-2 representation of p.

Remark 12. An algorithm for this problem would allow resolution of the existence of a Fermat prime $2^{2^k} + 1$ for $k > 4$.

2. Given a DFA M with input alphabet $\{0, 1\}$, decide if there exists at least one integer $n \geq 0$ such that M accepts the base-2 representation of n^2.

Acknowledgments. We thank Hendrik Jan Hoogeboom and the referees for their helpful comments.

References

1. Alur, R., Deshmukh, J.V.: Nondeterministic streaming string transducers. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 1–20. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22012-8_1
2. Bar-Hillel, Y., Perles, M., Shamir, E.: On formal properties of simple phrase structure grammars. Z. Phonetik. Sprachwiss. Kommuniationsforsch. **14**, 143–172 (1961)
3. Book, R.V., Otto, F.: String-Rewriting Systems. Springer, New York (1993). doi:10.1007/978-1-4613-9771-7
4. Cobham, A.: Uniform tag sequences. Math. Syst. Theor. **6**, 164–192 (1972)
5. Engelfriet, J., Rozenberg, G.: Fixed point languages, equality languages, and representation of recursively enumerable languages. J. Assoc. Comput. Mach. **27**, 499–518 (1980)
6. Ginsburg, S., Rose, G.F.: Some recursively unsolvable problems in ALGOL-like languages. J. Assoc. Comput. Mach. **10**, 29–47 (1963)
7. Hoogeboom, H.J.: Are there undecidable properties of non-turing-complete automata? Posting on stackexchange, 20 October 2012. http://cs.stackexchange.com/questions/1697/are-there-undecidable-properties-of-non-turing-complete-automata

8. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading (1979)
9. Lyndon, R.C., Schützenberger, M.P.: The equation $a^M = b^N c^P$ in a free group. Mich. Math. J. **9**, 289–298 (1962)
10. Post, E.: Absolutely unsolvable problems and relatively undecidable propositions: account of an anticipation. In: Davis, M. (ed.) The Undecidable, pp. 338–433. Raven Press, Hewlett (1965)
11. Rowland, E., Shallit, J.: Automatic sets of rational numbers. Int. J. Found. Comput. Sci. **26**, 343–365 (2015)
12. Rozenberg, G., Salomaa, A.: Cornerstones of Undecidability. Prentice-Hall, New York (1994)
13. Schaeffer, L., Shallit, J.: The critical exponent is computable for automatic sequences. Int. J. Found. Comput. Sci. **23**, 1611–1626 (2012)
14. Shallit, J.: A Second Course in Formal Languages and Automata Theory. Cambridge University Press, Cambridge (2009)
15. Shallit, J.O.: Numeration systems, linear recurrences, and regular sets. Inf. Comput. **113**, 331–347 (1994)
16. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proc. Lond. Math. Soc. **42**, 230–265 (1936)