The Simplest Language Where Equivalence of Finite Substitutions Is Undecidable

Michal Kunc *

Department of Mathematics and Statistics, Masaryk University,
Janáčkovo nám. 2a, 602 00 Brno, Czech Republic,
kunc@math.muni.cz, http://www.math.muni.cz/~kunc/

Abstract. We show that it is undecidable whether two finite substitutions agree on the binary language a^*b. This in particular means that equivalence of nondeterministic finite transducers is undecidable even for two-state transducers with unary input alphabet and whose all transitions start from the initial state.

1 Introduction

Existence of solutions of equations over words was proved decidable in a breakthrough paper of Makanin [12]. It is now also well known that solvability of word equations is decidable even for infinite rational systems of equations [2, 1, 5]. However, if we consider instead of equations over words equations over languages where the only operation is concatenation, the solvability problem becomes much more complicated.

If constants in equations are allowed to be any regular languages, existence of arbitrary solutions becomes undecidable already for very simple systems of equations [9]. But there is no such result about equations with only finite constants, and we also have virtually no knowledge about the solvability of finite systems of equations over finite or regular languages, i.e. where only finite or regular solutions are allowed. On the other hand, it is known that already for a very simple fixed rational system of such equations, it is even undecidable whether given finite languages form its solution. This can be equivalently formulated as undecidability of equivalence of two finite substitutions on a fixed regular language. Such a result was first proved for the regular language $a\{b,c\}^*d$ by Lisovik [11], and later improved to the language ab^*c in [8]. In this paper we prove that the same undecidability result actually holds even for the simplest language where the problem is not trivially decidable, namely for the language a^*b (note that for each language over a one-letter alphabet it is always sufficient to perform a certain fixed finite number of tests).

These results about finite substitutions can be also interpreted as undecidability of the equivalence problem for very restricted classes of finite transducers,

* Supported by the Grant no. 201/06/0936 of the Grant Agency of the Czech Republic.

continuing the long lasting search for such restrictions initiated in 1968 by the undecidability result of Griffiths [4] for general transducers (see [8] for a more comprehensive overview of related results). From this point of view, our result corresponds to transducers over unary input alphabet having only two states and no transitions starting from the final state. Therefore the result provides in this direction the smallest class where the equivalence problem is undecidable.

We assume the reader to be familiar with basic notions of formal language theory, which can be found for instance in [13]. As usual, we denote by X^* the set of all words arising by concatenating elements of X together with the empty word ε. The length of a word $w \in A^*$ over an alphabet A is written as $|w|$. The concatenation of two words $u, v \in A^*$ is denoted by uv. The operation of concatenation is extended to languages by the rule $KL = \{\, uv \mid u \in K,\ v \in L \,\}$. If some words $u, v, w \in A^*$ satisfy $uv = w$, then u and v are called a prefix and a suffix of w, respectively, and we write $u = wv^{-1}$ and $v = u^{-1}w$. We will also use the notation $A^{-1}L = \{\, w \in A^* \mid Aw \cap L \neq \emptyset \,\}$. And if $uvw = z$ for some $u, v, w, z \in A^*$, then v is called a factor of z.

2 Main Result

We are going to prove our result in the following form:

Theorem 1. *It is undecidable whether given three finite languages K, L, M satisfy $K^nM = L^nM$ for every non-negative integer n.*

This result can be easily translated to undecidability results for the problems of equivalence of finite substitutions on a^*b and equivalence of finite transducers. If we consider for finite languages K, L, M the substitutions $\varphi, \psi\colon \{a, b\}^* \to A^*$ defined by $\varphi(a) = K$, $\psi(a) = L$ and $\varphi(b) = \psi(b) = M$, then Theorem 1 can be directly reformulated in terms of deciding equivalence of finite substitutions as follows:

Corollary 1. *Equivalence of finite substitutions on the binary language a^*b is undecidable.*

As in [8], the language a^*b can be replaced by a two-state automaton with outputs of loops in the initial state defined according to K or L and outputs of transitions leading to the final state defined according to M; this way our result immediately implies undecidability of equivalence of finite transducers of a very special form.

Corollary 2. *It is undecidable whether given two nondeterministic two-state finite transducers with unary input alphabet, whose all transitions start from the initial state, are equivalent.*

3 Proof of the Result

The rest of the paper is devoted to proving Theorem 1. The proof generally follows the idea used to prove the analogous result in [8] for the equality $NK^nM = NL^nM$. We encode into our problem the universality problem for blind counter automata with all states final; these automata were first studied by Greibach [3], and the universality problem for the restricted class of blind counter automata we consider was proved undecidable by Lisovik [10] (see also [6]). A blind counter automaton consists of a nondeterministic finite automaton and one counter that can assume arbitrary integer values. Each transition of the automaton reads a letter from the input and possibly modifies the counter by either adding or subtracting one. No information about the current value of the counter is available to the automaton. The automaton accepts a given word if, starting from the initial state and zero-valued counter, it can read the word so that at the end the counter returns back to zero. We can assume that the automaton works over a binary alphabet $\{a, b\}$ (see [7, Corollary 1.2]).

Formally, such an automaton is a triple $\mathcal{S} = (S, 1, \delta)$, where $S = \{1, \ldots, s\}$ is its set of states, that we denote simply by numbers, 1 is the initial state, and $\delta = \{\mathbf{t}_1, \ldots, \mathbf{t}_t\} \subseteq S \times S \times \{a, b\} \times \{-1, 0, 1\}$ is the set of transitions, where a transition $\mathbf{t}_q = (i, j, x, k)$ starting from the state i and leading to the state j is labelled by x and has value k. Without loss of generality, we additionally assume that the initial state 1 is not reachable from the other states and that all transitions starting from the initial state have value zero. The set of all states reachable from a state i by a path labelled by a word $v \in \{a, b\}^*$ and having the sum of values of its transitions p will be denoted $\delta^*(i, v, p)$. The automaton accepts a word $v \in \{a, b\}^*$ if $\delta^*(1, v, 0) \neq \emptyset$.

In order to prove Theorem 1, we construct for each such automaton $\mathcal{S}$ finite languages K, L, M such that the automaton accepts all words if and only if these languages satisfy $K^nM = L^nM$ for every n.

The main difference from the construction in [8] is that we have to encode into the languages K and L not only transitions of the automaton and their labels and values, but also the initial state.

The words in K^nM encoding computations of $\mathcal{S}$ consist of many concatenated copies of the words $(wa)^3$ and $(wb)^3$, where w is a certain word containing only auxiliary letters distinct from a and b, which can be cut in different places to represent the current state of the computation or the currently performed transition. A word read by the automaton $\mathcal{S}$ represented by a given element of K^nM is obtained by replacing every copy of $(wa)^3$ by a and every copy of $(wb)^3$ by b. The values of transitions are encoded as the number of words from K needed to construct the corresponding copy of $(wa)^3$ or $(wb)^3$. For zero-valued transitions, the corresponding word is constructed using one element of K; for decrementing transitions, already one half of an element of K builds the whole word; and for incrementing transitions, one and a half element of K is needed.

In order to achieve this, all words in K and L will be in fact constructed by concatenating two words from a certain language L_1 of basic building blocks.

The language $L \supseteq K$ is constructed so that L^nM contains in addition to the words of K^nM exactly words obtained by concatenating n copies of words $(wa)^3$ and $(wb)^3$, so it allows to count the number of performed transitions and to compare it with the values obtained by accumulating the numbers during the production of a word in K^nM. These numbers are equal if and only if the sum of transition values of the computation corresponding to the word from K^nM is zero. Therefore the equality $K^nM = L^nM$ becomes true precisely if all words over $\{a, b\}$ correspond to some computation of $\mathcal{S}$ that resets the counter to zero.

This is achieved by constructing L by adding to the language K two words (the sublanguage J below) that start this counting. Since these initial words belong to L, they can actually occur anywhere in the power L^n, and therefore we need many auxiliary words in K for building words obtained in L^n by using words from J inside the product.

Finally, the language M serves for stopping the computation at any state by completing the currently unfinished copy of w.

The languages K, L, M will be defined over the alphabet

$$A = \{a, b, \#, \$, a_1, \dots, a_s, b_1, \dots, b_t, c_1, \dots, c_t, e, \bar{e}, f, \bar{f}, g, \bar{g}\}\,.$$

First, consider the word $w = \#a_1 \cdots a_s \$ e \bar{e} f \bar{f} g \bar{g} b_1 c_1 \cdots b_t c_t$, and note that every letter occurs only once in w. We denote by ${}_xw$ the suffix of w starting with $x \in A$. Let $w_x = w({}_xw)^{-1}$ be the corresponding prefix, and let ${}_xw_y = (w_x)^{-1}w({}_xw)^{-1}$ be the factor of w determined by letters $x, y \in A$. The fact that the automaton $\mathcal{S}$ is in a state i will be represented by cutting the word w right before the corresponding letter a_i. Similarly, we will cut w before b_q or c_q if the automaton is just performing the transition $\mathbf{t}_q$. Both letters b_q and c_q are needed to deal with transitions incrementing the counter (to separate the three blocks making up the corresponding copy of $(wa)^3$ or $(wb)^3$), only one of them is needed for zero-valued transitions, and none for decrementing ones.

Now we take the two words responsible for starting counting in L^n:

$$J = \{(wa)^2 w_e, (wb)^2 w_f\}$$

Then we define an auxiliary language L_0, which is a union of several languages defined below. The first part of L_0 consists of the two words from J and some additional words used to compensate these two words whenever they occur in L^n somewhere else than at the very beginning:

$$\{\#^{-1}, \varepsilon, \#, e, f, g\}(wa)^2 w_e \{\$^{-1}, \varepsilon, \$, \bar{e}, \bar{f}\} \cup$$
$$\{\#^{-1}, \varepsilon, \#, e, f, g\}(wb)^2 w_f \{\bar{e}^{-1}, \varepsilon, \$, \bar{e}, \bar{f}\} \subseteq L_0$$

The rest of counting is performed by means of the words

$$\{{}_ewaw_g, {}_fwbw_g, {}_gwawaw_e, {}_gwbwbw_f\} \subseteq L_0\,.$$

These words will be compensated by

$${}_\$waw_g\{\bar{f}^{-1}, \$, \bar{e}, \bar{f}\} \cup {}_{\bar{e}}wbw_g\{\bar{f}^{-1}, \$, \bar{e}, \bar{f}\} \cup$$
$${}_{\bar{f}}wawaw_e\{\$^{-1}, \$, \bar{e}, \bar{f}\} \cup {}_{\bar{f}}wbwbw_f\{\bar{e}^{-1}, \$, \bar{e}, \bar{f}\} \subseteq L_0\,.$$

Now we add to L_0 some words for every transition $\mathbf{t}_q = (i, j, x, k)$. If it is an initial transition, i.e. $i = 1$, then we add the words $(wx)^2 w_{b_q}$ and ${}_{b_q}wxw_{a_j}$. If this transition decrements the counter, i.e. $k = -1$, then we add only the word ${}_{a_i}wx(wx)^2 w_{a_j}$. If $k = 0$ and $i \neq 1$, we add to L_0 the words ${}_{a_i}wxwxw_{b_q}$ and ${}_{b_q}wxw_{a_j}$. And finally, if $k = 1$ then we add the words ${}_{a_i}wxw_{b_q}$, ${}_{b_q}wxw_{c_q}$ and ${}_{c_q}wxw_{a_j}$. This concludes the definition of L_0.

We will denote by F the language consisting of all factors of words from $\{wa, wb\}^*$. Let L_1 consist of

$$L_1' = {}_{\$}waw_g\{\#, e, f\} \cup {}_{\bar{e}}wbw_g\{\#, e, f\} \cup {}_{\bar{f}}wawaw_e\{\#, f, g\} \cup {}_{\bar{f}}wbwbw_f\{\#, e, g\}$$

and of all words of the form xuy, where $x \in \{\varepsilon, \$, \bar{e}, \bar{f}\}$, $u \in L_0$ and $y \in \{\varepsilon, \#, e, f, g\}$, such that either $u \notin F$ or

$$x \neq \varepsilon \implies xu \notin F \quad \text{and} \quad y \neq \varepsilon \implies uy \notin F\,.$$

We will also use the notations $K_0 = L_0 \setminus J$ and $K_1 = L_1 \setminus J$. Note that in particular $L_0 \subseteq L_1$ and $K_0 \subseteq K_1$.

Finally, define $K = K_1^2$, $L = L_1^2$ and

$$M = \{\varepsilon\} \cup \bigcup_{i \neq 1} \{{}_{a_i}w_g, \bar{f}_{a_i}w_g\}\,.$$

Claim 1. *For every $n \geq 1$, the languages K, L, M satisfy*

$$L^n M = K^n M \cup \{(wa)^3, (wb)^3\}^n w_g\,.$$

Proof. Since $L \subseteq K$, in order to prove the claim, it is enough to verify the inclusions

$$\{(wa)^3, (wb)^3\}^n w_g \subseteq L^n M\,, \tag{1}$$

$$L^n M \subseteq K^n M \cup \{(wa)^3, (wb)^3\}^n w_g\,. \tag{2}$$

The inclusion (1) follows by induction on n using the empty word from M and the formulas

$$(wa)^3 w_g = (wa)^2 w_e \cdot {}_e waw_g \in L\,,$$
$$(wb)^3 w_g = (wb)^2 w_f \cdot {}_f wbw_g \in L$$

for the basis of the induction and

$$\{(wa)^3, (wb)^3\}^n (wa)^3 w_g = \{(wa)^3, (wb)^3\}^n w_g \cdot {}_g wawaw_e \cdot {}_e waw_g \subseteq L^{n+1}\,,$$
$$\{(wa)^3, (wb)^3\}^n (wb)^3 w_g = \{(wa)^3, (wb)^3\}^n w_g \cdot {}_g wbwbw_f \cdot {}_f wbw_g \subseteq L^{n+1}$$

for the induction step.

In order to verify (2) we take any $2n+1$-element sequence σ belonging to

$$\underbrace{L_1 \times \cdots \times L_1}_{2n \text{ times}} \times M$$

whose concatenation does not belong to $\{(wa)^3,(wb)^3\}^n w_g$, and successively modify σ to replace all words from J in this sequence by words from K_1 without changing neither the length of the sequence nor the resulting concatenation. We distinguish several cases according to the word in σ directly preceding the word from J. When writing parts of σ, we separate neighbouring words in the sequence by means of the multiplication sign.

Let us start with the case of a word from J preceded by a word from L_1'. Then the word from J can be replaced using one of the following rules:

$$\begin{aligned}
{}_{\$}waw_g\{\#,e,f\}\cdot J &= {}_{\$}waw_{\bar f}\cdot \bar f\{\#,e,f\}J \subseteq K_0\cdot K_1\\
{}_{\bar e}wbw_g\{\#,e,f\}\cdot J &= {}_{\bar e}wbw_{\bar f}\cdot \bar f\{\#,e,f\}J \subseteq K_0\cdot K_1\\
{}_{\bar f}wawaw_e\{\#,f,g\}\cdot J &= {}_{\bar f}wawaw_{\$}\cdot \$\{\#,f,g\}J \subseteq K_0\cdot K_1\\
{}_{\bar f}wbwbw_f\{\#,e,g\}\cdot J &= {}_{\bar f}wbwbw_{\bar e}\cdot \bar e\{\#,e,g\}J \subseteq K_0\cdot K_1\,.
\end{aligned}$$

Next we deal with words from L_1 with $y=\varepsilon$, i.e. of the form xu, where $x\in\{\varepsilon,\$,\bar e,\bar f\}$ and $u\in L_0$. First, observe that no word from L_0 ends on a or b, and therefore $xu\#\in K_1$. This allows us to replace any word from J following xu by means of the inclusion

$$xu\cdot J = xu\#\cdot \#^{-1}J \subseteq K_1\cdot K_0\,.$$

Now consider the other words from L_1, i.e. of the form xuy, where $x\in\{\varepsilon,\$,\bar e,\bar f\}$, $u\in L_0$ and $y\in\{\#,e,f,g\}$. In this case we have $xu\in L_1$. If $xu\in K_1$ then we can use

$$xuy\cdot J\subseteq K_1\cdot K_0\,.$$

And if $xu\in J$ then $xu\in K_0\{\$,\bar e\}$ and from the inclusion $yJ\subseteq L_0\setminus F$ we obtain

$$xuy\cdot J\subseteq K_0\{\$,\bar e\}(L_0\setminus F)\subseteq K_0\cdot K_1\,.$$

It remains to deal with a word from J that is the first word of σ. We can assume that it is already the only word from J in σ. Let us take the longest initial part ρ of σ which is also an initial part of some of the sequences of the form

$$\{(wa)^2w_e\cdot {}_ewaw_g,(wb)^2w_f\cdot {}_fwbw_g\}\cdot\{{}_gwawaw_e\cdot {}_ewaw_g,{}_gwbwbw_f\cdot {}_fwbw_g\}^*\,.$$

If the length of ρ is greater than one, we replace every word belonging to ρ except for the last one according to the following rules: $(wa)^2w_e$ by $(wa)^2w_{\$}\in K_0$, $(wb)^2w_f$ by $(wb)^2w_{\bar e}\in K_0$, ${}_ewaw_g$ by ${}_{\$}waw_{\bar f}\in K_0$, ${}_fwbw_g$ by ${}_{\bar e}wbw_{\bar f}\in K_0$, ${}_gwawaw_e$ by ${}_{\bar f}wawaw_{\$}\in K_0$ and ${}_gwbwbw_f$ by ${}_{\bar f}wbwbw_{\bar e}\in K_0$. Observe that this modification does not affect the concatenation of these words of ρ except for losing one letter at the end, which is one of $\$$, $\bar e$ and $\bar f$, depending on what the last word of ρ is. This letter together with the last word of ρ forms one of the following words, that remain to be dealt with: ${}_{\$}waw_g$, ${}_{\bar e}wbw_g$, ${}_{\bar f}wawaw_e$ and ${}_{\bar f}wbwbw_f$. If the sequence ρ consists of only one word, then we have to deal with

this sole word, which belongs to J. Let us denote this remainder of ρ in both cases by r, and let v be the word that follows r in σ.

First assume that the length of ρ is maximal possible, i.e. equal to $2n$. Then $v \in M$ is the last word of σ and the remainder r is either ${}_{\$}waw_g$ or ${}_{\bar{e}}wbw_g$. The last word in σ cannot be the empty word $\varepsilon \in M$, since then its concatenation would belong to $\{(wa)^3, (wb)^3\}^n w_g$. If the last word in σ is ${}_{a_i}w_g \in M$, then it is sufficient to add to v the redundant letter $\bar{f}$ of r to obtain the word $\bar{f}_{a_i}w_g \in M$. And if the last word in σ is $\bar{f}_{a_i}w_g \in M$, then we replace r by one of the words ${}_{\$}waw_g\bar{f} \in K_0$ and ${}_{\bar{e}}wbw_g\bar{f} \in K_0$, and v by the word ${}_{a_i}w_g \in M$.

Now assume that the length of ρ is smaller than $2n$. If $v \in L_1'$ then $r \cdot v$ can be replaced by a sequence from $r\{\$, \bar{e}, \bar{f}\} \cdot A^{-1}L_1' \subseteq K_0 \cdot K_1$. Otherwise, we have $v = xuy \in K_1$. If $x \in \{\$, \bar{e}, \bar{f}\}$ then $uy \in L_1$ and we distinguish two cases. For $uy \in K_1$, we replace $r \cdot v$ by $rx \cdot uy \in K_0 \cdot K_1$. And for $uy \in J$, we replace it by $rx\# \cdot \#^{-1}uy \in K_1 \cdot K_0$.

It remains to consider words v where $x = \varepsilon$. In this case we replace r in the same way as the other words of ρ and denote by z the last letter of r, which is again one of $\$$, $\bar{e}$ and $\bar{f}$. It remains to verify that the word zv belongs to K_1, so that we can use it instead of v. This is certainly true if zv does not start with one of the pairs $\$e$, $\bar{e}f$ and $\bar{f}g$, since then we can take the letter z for x. So assume that zv does start with one of these pairs. Observe that v cannot be one of the words ${}_{e}waw_g$, ${}_{f}wbw_g$, ${}_{g}wawaw_e$ and ${}_{g}wbwbw_f$ because of the maximality of ρ. Therefore we have either

$$v \in {}_{e}waw_g\{\#, e, f\} \cup {}_{f}wbw_g\{\#, e, f\} \cup {}_{g}wawaw_e\{\#, f, g\} \cup {}_{g}wbwbw_f\{\#, e, g\}\,,$$

which means that $zv \in L_1' \subseteq K_1$, or

$$u \in \{e, f, g\}(wa)^2w_e\{\$^{-1}, \varepsilon, \$, \bar{e}, \bar{f}\} \cup \{e, f, g\}(wb)^2w_f\{\bar{e}^{-1}, \varepsilon, \$, \bar{e}, \bar{f}\}\,,$$

in which case $u \notin F$, and so z can be taken for x. □

To describe which words from $\{(wa)^3, (wb)^3\}^n w_g$ belong to K^nM we will use the injective homomorphism $\varphi\colon \{a, b\}^* \to A^*$ defined by the rule $\varphi(x) = (wx)^3$ for $x \in \{a, b\}$.

Claim 2. *For every $n \geq 1$, the languages K, M satisfy*

$$K^nM \cap \{(wa)^3, (wb)^3\}^n w_g = \{\, \varphi(v)w_g \mid v \in \{a, b\}^n,\ \delta^*(1, v, 0) \neq \emptyset \,\}\,.$$

Proof. In order to prove the converse inclusion, we show by induction with respect to the length of a word $v \in \{a, b\}^+$ that if $i \in \delta^*(1, v, p)$ for some state i and some integer p, then

$$\varphi(v)w_{a_i} \in K_0^{2|v|+p}\,. \tag{3}$$

Starting with words of length one, take any $x \in \{a, b\}$ and any transition from the initial state labelled by x, which is by our initial assumption of the form $\mathbf{t}_q = (1, j, x, 0)$. Then

$$\varphi(x)w_{a_j} = (wx)^2w_{b_q} \cdot {}_{b_q}wxw_{a_j} \in K_0^2\,.$$

Now assume that (3) is true for some v, i and p, and consider any transition starting from i, i.e. of the form $\mathbf{t}_q = (i, j, x, k)$ with $x \in \{a, b\}$ and $k \in \{-1, 0, 1\}$. If $k = -1$ then

$$\varphi(vx)w_{a_j} = \varphi(v)w_{a_i} \cdot {}_{a_i}wx(wx)^2 w_{a_j} \in K_0^{2|vx|+(p-1)}\,.$$

For $k = 0$, we have $i \neq 1$ since the initial state is not reachable by v, and therefore

$$\varphi(vx)w_{a_j} = \varphi(v)w_{a_i} \cdot {}_{a_i}wxwxw_{b_q} \cdot {}_{b_q}wxw_{a_j} \in K_0^{2|vx|+p}\,.$$

And finally, if $k = 1$ then

$$\varphi(vx)w_{a_j} = \varphi(v)w_{a_i} \cdot {}_{a_i}wxw_{b_q} \cdot {}_{b_q}wxw_{c_q} \cdot {}_{c_q}wxw_{a_j} \in K_0^{2|vx|+(p+1)}\,.$$

This proves the induction step.

If we now take $v \in \{a, b\}^n$ such that $i \in \delta^*(1, v, 0)$, then (3) gives us $\varphi(v)w_{a_i} \in K^n$, which implies

$$\varphi(v)w_g = \varphi(v)w_{a_i} \cdot {}_{a_i}w_g \in K^n \cdot M\,,$$

as required.

Now we turn to the direct inclusion. Since all factors of words from the set $\{(wa)^3, (wb)^3\}^n w_g$ are in F, it is enough to consider only words from K and M which belong to F. Observe that all words in $K \cap F$ clearly belong to $(K_0 \cap F)^2$, and that the language $K_0 \cap F$ consists of the words

$$\begin{gathered}{}_{a_1}wawaw_{\$},\ {}_{a_1}wawaw_{e},\ (wa)^2w_{\$},\ {}_{a_1}wbwbw_{\bar{e}},\ {}_{a_1}wbwbw_{f},\ (wb)^2w_{\bar{e}},\\ {}_{e}waw_{g},\ {}_{f}wbw_{g},\ {}_{g}wawaw_{e},\ {}_{g}wbwbw_{f},\ {}_{\$}waw_{\bar{f}},\ {}_{\bar{e}}wbw_{\bar{f}},\ {}_{\bar{f}}wawaw_{\$},\ {}_{\bar{f}}wbwbw_{\bar{e}}\end{gathered}$$

and additionally all words corresponding to the transitions of the automaton $\mathcal{S}$. It is easy to verify by induction on m that every word $u \in K_1^m$, which is a prefix of a word from $\{wa, wb\}^*$, actually belongs to one of the following sets:

$$\begin{gathered}\{wa, wb\}^*\{w_{\$}, w_{\bar{e}}, w_{\bar{f}}\}\\ \{\,\varphi(v)w_{a_i} \mid i \in \delta^*(1, v, p),\ m = 2|v| + p\,\}\\ \{\,\varphi(v)(wx)^2w_{b_q} \mid i \in \delta^*(1, v, p),\ \mathbf{t}_q = (i, j, x, 0) \in \delta,\ m = 2|v| + p + 1\,\}\\ \{\,\varphi(v)wxw_{b_q} \mid i \in \delta^*(1, v, p),\ \mathbf{t}_q = (i, j, x, 1) \in \delta,\ m = 2|v| + p + 1\,\}\\ \{\,\varphi(v)(wx)^2w_{c_q} \mid i \in \delta^*(1, v, p),\ \mathbf{t}_q = (i, j, x, 1) \in \delta,\ m = 2|v| + p + 2\,\}\end{gathered}$$

This in particular shows that u cannot belong to the language $\{(wa)^3, (wb)^3\}^n w_g$. Since $M \cap F$ contains, apart from the empty word, only words ${}_{a_i}w_g$, every element of $K^nM \cap \{wa, wb\}^* w_g$ belongs to the set

$$\{\,\varphi(v)w_{a_i} \cdot {}_{a_i}w_g \mid i \in \delta^*(1, v, p),\ 2n = 2|v| + p\,\}\,.$$

This finally gives

$$\begin{aligned}K^nM \cap \{(wa)^3, (wb)^3\}^n w_g &\subseteq \{\,\varphi(v)w_g \mid \delta^*(1, v, p) \neq \emptyset,\ 2n = 2|v| + p,\ |v| = n\,\}\\ &\subseteq \{\,\varphi(v)w_g \mid v \in \{a, b\}^n,\ \delta^*(1, v, 0) \neq \emptyset\,\}\,,\end{aligned}$$

which concludes the proof of the claim. □

From these two claims we can now easily prove the main result.

Proof (of Theorem 1). It is sufficient to show that $\mathcal{S}$ accepts all words from $\{a,b\}^+$ if and only if the languages K, L, M constructed from $\mathcal{S}$ satisfy $K^nM = L^nM$ for every n. First assume that $\mathcal{S}$ accepts all words. Then Claim 2 gives us

$$K^nM \cap \{(wa)^3,(wb)^3\}^n w_g = \varphi(\{a,b\}^n)w_g = \{(wa)^3,(wb)^3\}^n w_g\,,$$

and therefore $K^nM = L^nM$ by Claim 1. Conversely, if $K^nM = L^nM$ then Claim 1 implies $\{(wa)^3,(wb)^3\}^n w_g \subseteq K^nM$, which turns Claim 2 into

$$\{\,\varphi(v)w_g \mid v \in \{a,b\}^n,\ \delta^*(1,v,0) \neq \emptyset\,\} = \{(wa)^3,(wb)^3\}^n w_g\,,$$

showing $\delta^*(1,v,0) \neq \emptyset$ for every word $v \in \{a,b\}^n$. This proves that every word in $\{a,b\}^+$ is accepted by $\mathcal{S}$. □

References

1. Albert, M.H., Lawrence, J.: A proof of Ehrenfeucht's conjecture. *Theoret. Comput. Sci.* **41**(1) (1985), 121–123.
2. Culik II, K., Karhumäki, J.: Systems of equations over a free monoid and Ehrenfeucht's conjecture. *Discrete Math.* **43**(2-3) (1983), 139–153.
3. Greibach, S.A.: Remarks on blind and partially blind one-way multicounter machines. *Theoret. Comput. Sci.* **7**(3) (1978), 311–324.
4. Griffiths, T.V.: The unsolvability of the equivalence problem for Λ-free nondeterministic generalized machines. *J. Assoc. Comput. Mach.* **15** (1968), 409–413.
5. Guba, V.S.: Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems. *Mat. Zametki* **40**(3) (1986), 321–324.
6. Halava, V., Harju, T.: Undecidability in integer weighted finite automata. *Fund. Inform.* **38**(1-2) (1999), 189–200.
7. Halava, V., Harju, T.: Undecidability of the equivalence of finite substitutions on regular language. *Theor. Inform. Appl.* **33** (1999), 117–124.
8. Karhumäki, J., Lisovik, L.P.: The equivalence problem of finite substitutions on ab^*c, with applications. *Internat. J. Found. Comput. Sci.* **14**(4) (2003), 699–710.
9. Kunc, M.: The power of commuting with finite sets of words. To appear in *Theory Comput. Syst.*
10. Lisovik, L.P.: An undecidable problem for countable Markov chains. *Kibernetika* (2) (1991), 1–6.
11. Lisovik, L.P.: The equivalence problem for finite substitutions on regular languages. *Dokl. Akad. Nauk* **357**(3) (1997), 299–301.
12. Makanin, G.S.: The problem of solvability of equations in a free semigroup. *Mat. Sb.* **103**(2) (1977), 147–236.
13. Rozenberg, G., Salomaa, A. (eds.): *Handbook of Formal Languages.* Springer, Berlin (1997).