Discrete Mathematics and Theoretical Computer Science DMTCS vol. **12:4**, 2010, 9–18

Noneffective Regularity of Equality Languages and Bounded Delay Morphisms[†]

Juhani Karhumäki and Aleksi Saarela

Department of Mathematics and Turku Centre for Computer Science TUCS, University of Turku, Turku, Finland

received 30th October 2009, accepted 1st June 2010.

We give an instance of a class of morphisms for which it is easy to prove that their equality set is regular, but its emptiness is still undecidable. The class is that of bounded delay 2 morphisms.

Keywords: regularity, bounded delay, Post correspondence problem

1 Introduction

Regular (or rational) languages constitute a fundamental class of formal languages, which is typically viewed as a simple, or even the simplest class of languages. For example, decision problems on regular languages are – with a few exceptions – algorithmically decidable.

However, there also exists another viewpoint. There are some questions, which lead to regular languages, but the mechanism is so complicated that the regularity holds noneffectively (the meaning of this assertion will be explained below). A splendid example is a result of Haines (or, in fact, that of Higman) stating that the upper closure of any language under the relation of being a sparse subword is regular, see e.g [Hai69]. Another example of this nature is the regularity of equality languages of bounded delay morphisms, which is the topic of this paper. Our goal is to give a simple – a textbook type – proof of this fundamental result.

Equality languages are sets of solutions of instances of the Post Correspondence Problem (PCP for short). Hence their emptiness is undecidable as was proved by E. Post already in 1946, see [Pos46]. Later it was revealed that their generating power is very high: each recursively enumerable language is obtained as a morphic image of an intersection of an equality language and a regular language, see e.g. [Cul79] or [Kar09] for a recent survey. On the other hand, certain equality languages, namely those with elementary morphisms, played a crucial role in a nice solution of the celebrated D0L sequence equivalence problem, see [ER78]. For these morphisms the equality language is regular. A further analysis revealed that the regularity was, in fact, a consequence of so-called bounded delay property, see [CK85] and [HK97].

Interesting problems arose: Is the equality language of bounded delay morphisms effectively regular? Is the PCP decidable for bounded delay morphisms? These problems were answered negatively by K.

[†]Supported by the Academy of Finland under grant 121419

Ruohonen in [Ruo85]. In fact, he proved his result in a stronger form of biprefix morphisms. A drawback of his proof is that it is quite complicated. It relies, as an earlier proof showing the undecidability of PCP for injective morphisms, see [Lec63], on reversible Turing machines. However, an equality set of injective morphisms need not be regular, see [KS79].

Our goal here is to give an instance of a class of morphisms for which the PCP is undecidable, and the equality languages are regular, and moreover the proofs of these results are short. Our family is that of morphisms with bounded delay 2. Our proofs use the ideas of [Ruo85], but allow simpler constructions, and consequently shorter proofs.

We will briefly outline the structure and constructions in this paper. We use the halting problem of Turing machines on empty input as our undecidable problem. We want to simulate their computations by the equality mechanism of two cooperating morphisms. It is well known that this can be done. However, the morphisms need not possess a bounded delay property. A key idea is here that instead of using normal configurations of Turing machines we use so-called extended configurations which contain some extra local information. This makes it possible to do the above simulation by bounded delay morphisms. The whole approach does not require the use of so-called reversible Turing machines, as was crucial in [Ruo85], although some ingredients of reversible Turing machines motivate the definition of extended configurations. As a consequence we obtain a direct and relatively simple proof for the undecidability of PCP for bounded delay morphisms.

2 Notions

We assume that the reader is familiar with the basics of automata theory and combinatorics on words, see e.g. [HU79], [Lot02] and [CK97]. For more on morphisms of free monoids we refer to [HK97]. We only recall here that the notation $u \leq v$ is used to specify that word u is a prefix of word v, and that words u and v are *comparable* if $u \leq v$ or $v \leq u$.

The *equality language* of two morphisms $f, g : \Sigma^* \to \Gamma^*$ is the set

$$E(f, g) = \{w \in \Sigma^+ : f(w) = g(w)\} . \tag{1}$$

The PCP asks whether this language is empty or not. The modified PCP asks whether the language $E(f, g) \cap a\Sigma^*$ is empty for a given letter $a \in \Sigma$.

A morphism $h : \Sigma^* \to \Gamma^*$ is of *bounded delay* n, if there are no letters $a, b \in \Sigma$ and words $u, v \in \Sigma^n$, for which $a \neq b$ and $h(au) \leq h(bv)$. Further it is of *bounded delay* if it is so for some n.

An *instantaneous description* (ID) of a Turing machine (TM) at some point of its computation is $u(q, a)v$, where q is the current state, a is the symbol under the head, and u and v are the parts to the left and to the right of it. So every computation has a sequence of associated IDs, each of which describes the configuration of the machine at one step of the computation.

The article [Ruo85] contains two constructions of TMs: First, for an arbitrary TM M an equivalent reversible machine M' is constructed. Second, an equivalent so-called synchronized modulo 2 machine M'' is constructed for M'. We do not need these constructions. First reversible TMs were constructed already in 1970s, see [Ben73], but their importance has increased after the invention of quantum computing, see [Hir03].

3 Extended Instantaneous Descriptions

Our goal is to prove the undecidability of the PCP for bounded delay morphisms. We will do this by reducing it to the halting problem of TMs on empty input. More precisely, for an arbitrary TM we will construct two bounded delay morphisms such that their equality set contains a word starting with a specific letter if and only if the machine accepts the empty word.

In a standard proof of the undecidability of the PCP a computation of a TM is simulated with two morphisms, which operate on IDs (see e.g. [Kar09]). We wish to do this simulation with bounded delay morphisms. One of the problems in this approach is that a bounded delay morphism is necessarily injective on letters, but the transition function of a TM can be non-injective. This is why we will in this section define extended instantaneous descriptions. In the next section we will then define the bounded delay morphisms, which will operate on these extended configurations, and prove the results.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, *, \{h\})$ be a deterministic TM, where Q is the set of states, Σ is the input alphabet, Γ is the tape alphabet, δ is the (partial) transition function, q_0 is the initial state, $*$ is the blank symbol and h is the halting state. For convenience, we assume that M never enters q_0, always halts in h and never writes $*$.

Example 3.1 *Let $M = (\{q_0, h\}, \{a\}, \{a, *\}, \delta, q_0, *, \{h\})$, where $\delta(q_0, *) = (h, a, R)$. If the input is empty, then M will write a, move to the right and halt. The IDs of this computation are $(q_0, *)$ and $a(h, *)$.*

Let $\Delta = \Gamma \cup (Q \times \Gamma)$. Let $\Theta = \Delta \times (\{S\} \cup (Q \times \Gamma))$, where S is a new symbol. We write the second component of an element of Θ below the first component. For example, if $q \in Q$ and $a \in \Gamma$, then

$$\underset{S}{(q, a)},\ \underset{(q,a)}{a} \in \Theta. \tag{2}$$

We define *extended instantaneous descriptions* (EID) of the computation of M with empty input. Every EID will be an element of $(Q \times \Gamma)^* \# \Gamma^* \Theta \Delta^* \$$, where $\#$ and $\$$ are new symbols. The initial EID is

$$\beta_0 = \#\underset{S}{(q_0, *)}\$. \tag{3}$$

The next rows define the rules $\beta_i \to \beta_{i+1}$ with which we can derive the next EID β_{i+1} from an EID β_i. Here $a, b, c, b' \in \Gamma$, $a, c, b' \neq *$, $q, q' \in Q$, $y, z \in \Delta$, $u_1 \in (Q \times \Gamma)^*$, $u_2 \in \Gamma^*$ and $u_3 \in \Delta^*$.

$$u_1\#u_2\underset{S}{a}zu_3\$ \to u_1\#u_2a\underset{S}{z}u_3\$, \tag{4}$$

$$u_1\#u_2a\underset{S}{(q,b)}u_3\$ \to u_1\#u_2\underset{(q,b)}{(q',a)}b'u_3\$, \qquad \text{if } \delta(q,b) = (q',b',L), \tag{5}$$

$$u_1\#\underset{S}{(q,b)}u_3\$ \to u_1\#\underset{(q,b)}{(q',*)}b'u_3\$, \qquad \text{if } \delta(q,b) = (q',b',L), \tag{6}$$

$$u_1\#u_2\underset{S}{(q,b)}cu_3\$ \to u_1\#u_2b'\underset{(q,b)}{(q',c)}u_3\$, \qquad \text{if } \delta(q,b) = (q',b',R), \tag{7}$$

$$u_1\#u_2\underset{S}{(q,b)}\$ \to u_1\#u_2b'\underset{(q,b)}{(q',*)}\$, \qquad \text{if } \delta(q,b) = (q',b',R), \tag{8}$$

$$u_1\#u_2a\underset{(q,b)}{y}u_3\$ \to u_1\#u_2\underset{(q,b)}{a}yu_3\$, \tag{9}$$

$$u_1\# \underset{(q,b)}{y}\ u_3\$ \to u_1(q,b)\#\underset{S}{y}u_3\$. \tag{10}$$

So if $\beta_i \to \beta_{i+1}$ for $i = 0, 1, 2, \ldots$, then $\beta_0, \beta_1, \beta_2, \ldots$ are the EIDs. This sequence of EIDs can be finite or infinite. It can be easily seen that for every EID β_i there is at most one EID β_{i+1} such that $\beta_i \to \beta_{i+1}$, so the sequence is unique.

The idea of EIDs is that the part to the left of $\#$ contains the state-symbol-pairs of the computation up to some point, and the part between $\#$ and \$ becomes an ID, if the element of Θ is replaced with its first component. It is important that two consecutive EIDs differ only in one place. This is why the element of Θ moves back and forth and carries information in its second component. Lemma 3.3 and its proof explain how EIDs work.

Example 3.2 *Let M be the TM of Example 3.1. The EIDs are*

$$\begin{aligned}&\#\underset{S}{(q_0,*)}\$, \quad \#a\underset{(q_0,*)}{(h,*)}\$, \quad \#\underset{(q_0,*)}{a}(h,*)\$,\\ &(q_0,*)\#\underset{S}{a}(h,*)\$, \quad (q_0,*)\#a\underset{S}{(h,*)}\$.\end{aligned} \tag{11}$$

The next lemma shows the relation between IDs and EIDs.

Lemma 3.3 *Let $\alpha_0, \alpha_1, \alpha_2, \ldots$ be the IDs and $\beta_0, \beta_1, \beta_2, \ldots$ the EIDs of the computation of M with empty input. Let $\alpha_i = u_i x_i v_i$, where $u_i, v_i \in \Gamma^*$ and $x_i \in Q \times \Gamma$. Now there are $0 = j_0 < j_1 < j_2 < \ldots$ such that*

$$\beta_{j_i} = x_0 \ldots x_{i-1}\#u_i\underset{S}{x_i}v_i\$. \tag{12}$$

Proof: Clearly (12) holds for $i = 0$. We assume that it holds for i and prove that

$$\beta_{j_i} \to^+ x_0 \ldots x_i\#u_{i+1}\underset{S}{x_{i+1}}v_{i+1}\$, \tag{13}$$

where $\to^+$ is the transitive closure of $\to$. (Naturally we assume here that α_i is not the last ID, that is $\delta(x_i)$ and thus α_{i+1} are defined.)

By using one of the rules (5)–(8), we simulate one step of the computation and end up with the EID

$$\beta_{j_i+1} = x_0 \ldots x_{i-1}\#u_{i+1}\underset{x_i}{x_{i+1}}v_{i+1}\$. \tag{14}$$

By using rule (9) sufficiently many times and then rule (10), we get the EID

$$x_0 \ldots x_i\#\underset{S}{y}v\$, \tag{15}$$

where $yv = \alpha_{i+1}$. By using rule (4) sufficiently many times we get the required EID $\beta_{j_{i+1}}$. □

4 Morphisms

Like in the previous section, we will consider a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, *, \{h\})$, and we will use the notation, $\Delta = \Gamma \cup (Q \times \Gamma)$, $\Delta_1 = \Delta \cup \{\#, \$\}$ and $\Theta = \Delta \times (\{S\} \cup (Q \times \Gamma))$.

We will define two bounded delay morphisms $g_1, f_1 : A^* \to B^*$, which will simulate the computation of M with empty input. The alphabet B is the same as that used by EIDs, that is $B = \Delta_1 \cup \Theta$. For every $u \in B^*$ we define a letter $[u]$. The alphabet A will consist of a finite number of these letters. The morphism g_1 is defined so that $g_1([u]) = u$. The first column of Table 1 gives the values of g_1 on the letters of A, and thus it also implicitly defines the alphabet A. The second column gives the values of f_1 on these letters. In the table $a, b, c, b' \in \Gamma$, $a, c, b' \neq *$, $q, q' \in Q$, $x \in \Gamma \cup \{\#\}$, $y \in \Delta$ and $z \in \Delta_1$. There is a correspondence between the rows of the table and the rules (4)–(10) in the definition of EIDs.

Tab. 1: Morphisms g_1 and f_1

$u = g_1([u])$	$f_1([u])$	
z	z	
$x\underset{S}{a}z$	$xa\underset{S}{z}$	$z \neq (q_0, *), z \in \Delta$
$a\underset{S}{(q,b)}z$	$\underset{(q,b)}{(q',a)}b'z$	$\delta(q,b) = (q', b', L)$
$\#\underset{S}{(q,b)}z$	$\#\underset{(q,b)}{(q',*)}b'z$	$\delta(q,b) = (q', b', L)$
$x\underset{S}{(q,b)}c$	$xb'\underset{(q,b)}{(q',c)}$	$\delta(q,b) = (q', b', R)$
$x\underset{S}{(q,b)}\$$	$xb'\underset{(q,b)}{(q',*)}\$$	$\delta(q,b) = (q', b', R)$
$a \ \underset{(q,b)}{y} \ z$	$\underset{(q,b)}{a} \ yz$	
$\# \ \underset{(q,b)}{y} \ z$	$(q,b)\#y\underset{S}{z}$	$y \neq (q_0, *)$

The next lemma shows that g_1 and f_1 simulate M in the sense that if g_1 "reads" an EID, then f_1 "writes" the next EID.

Lemma 4.1 *Let β_j, β_{j+1} be two consecutive EIDs. There is a word $u \in A^*$ such that $g_1(u) = \beta_j$ and $f_1(u) = \beta_{j+1}$. Further, if $\beta_j \leq g_1(v)$ or $\beta_{j+1} \leq f_1(v)$, then $u \leq v$.*

Proof: Let $\beta_j = x_1 \dots x_m X y_1 \dots y_n$, where $x_i, y_i \in \Delta_1$ and $X \in \Theta$. If $\beta_j \leq g_1(v)$, then v must begin with $u = [x_1] \dots [x_{m-1}][x_m X y_1][y_2] \dots [y_n]$. Now $g_1(u) = \beta_j$. By comparing the rules (4)–(10) with Table 1 it can be verified that $f_1(u) = \beta_{j+1}$. If $\beta_{j+1} \leq f_1(v)$, then similarly it can be seen that v begins with $u = [x_1] \dots [x_{m-1}][x_m X y_1][y_2] \dots [y_n]$. □

The morphisms f_1 and g_1 are almost sufficient for simulating M, but we must extend them to handle initialization and termination. Initialization is easy, because there is only one initial EID, but termination is hard, because there are many potential final EIDs. This problem is handled by running the simulation backwards, which requires more complicated notation. The reversibility of a computation, which is enabled by the use of EIDs instead of IDs, is implicitly present here, as well as later in the proof of Lemma 4.4.

For some elements $x \in \Theta$ we need a separate copy of x. This will be denoted by $\overline{x}$. For some other elements x we want x and $\overline{x}$ to be equal. So we define a mapping $x \mapsto \overline{x}$ as follows: If $x = (h \underset{S}{,} a)$, where $a \in \Gamma$, let $\overline{x} = x$. For every other $x \in \Theta$, let $\overline{x}$ be a copy of x. Further, for every $x \in \Delta_1$, let $\overline{x} = x$. Finally, let $\overline{B} = \{\overline{x} : x \in B\}$ and $\overline{\Theta} = \{\overline{x} : x \in \Theta\}$. We extend the mapping $x \mapsto \overline{x}$ to a morphism $B^* \to (B \cup \overline{B})^*$. Note that

$$B \cap \overline{B} = \Delta_1 \cup \left\{ (h \underset{S}{,} a) : a \in \Gamma \right\}. \tag{16}$$

If $t = [u]$, where $u \in B^*$, let $\overline{t} = [\overline{u}]$. Let $\overline{A} = \{\overline{t} : t \in A\}$. We extend also this mapping $[u] \mapsto \overline{[u]}$ to a morphism $A^* \to (A \cup \overline{A})^*$. Note that $A \cap \overline{A} = \{[x] : x \in \Delta_1\}$.

We define two bounded delay morphisms $g, f : (A \cup \overline{A} \cup \{I, T\})^* \to (B \cup \overline{B} \cup \{I, T\})^*$:

$$g(t) = \begin{cases} g_1(t) & \text{if } t \in A, \\ \overline{f_1(s)} & \text{if } t = \overline{s} \in \overline{A}, \\ I & \text{if } t = I, \\ \#\overline{(q_0 \underset{S}{,} *)}\$T & \text{if } t = T, \end{cases} \qquad f(t) = \begin{cases} f_1(t) & \text{if } t \in A, \\ \overline{g_1(s)} & \text{if } t = \overline{s} \in \overline{A}, \\ I\#(q_0 \underset{S}{,} *)\$ & \text{if } t = I, \\ T & \text{if } t = T. \end{cases} \tag{17}$$

Here I and T are new symbols. If $t \in A \cap \overline{A}$, then $g_1(t) = \overline{f_1(t)}$ and $f_1(t) = \overline{g_1(t)}$, so f and g are well defined.

Lemma 4.2 *If M does not accept the empty word, then the set $\{w \in E(f, g) : I \leq w\}$ is empty, and if M does accept the empty word, then the set contains a unique minimal word (minimal with respect to the prefix ordering).*

Proof: Let the EIDs of M be $\beta_0, \beta_1, \beta_2, \ldots$. The only way to start is with $f(I)$ and $g(I)$. Then $f(I) = g(I)\beta_0$. By Lemma 4.1, if $f(Iw) = g(Iw)\beta_k$ and $\beta_k \to \beta_{k+1}$, then we must continue with $g(u_k) = \beta_k$ and $f(u_k) = \beta_{k+1}$, where $u_k \in A^*$.

If the computation of M does not halt, this will go on forever, and $f(Iw)$ and $g(Iw)$ can never match. If the computation stops in a nonhalting state, then there is no way to continue. So if M does not accept the empty word, then $f(Iw) \neq g(Iw)$ for all w.

If the computation halts and β_n is the last EID, then $f(Iu_0 \ldots u_{n-1}) = g(Iu_0 \ldots u_{n-1})\beta_n$ and $\beta_n = \overline{\beta}_n$. By Lemma 4.1, if $f(Iw) = g(Iw)\overline{\beta}_{k+1}$ and $\beta_k \to \beta_{k+1}$, then we must continue with $g(\overline{u}_k) = \overline{\beta}_{k+1}$ and $f(\overline{u}_k) = \overline{\beta}_k$.

So if $w = u_0 \ldots u_{n-1}\overline{u}_{n-1} \ldots \overline{u}_0$, then $f(Iw) = g(Iw)\overline{\beta}_0$, and IwT is the unique minimal word such that $f(IwT) = g(IwT)$. $\square$

Example 4.3 *Let M be the TM of Examples 3.1 and 3.2. Now*

$$\begin{aligned} g(w) = f(w) = {} & I\#(q_0 \underset{S}{,} *)\$\#a\underset{(q_0,*)}{(h, *)}\$\#\underset{(q_0,*)}{a}(h, *)\$ \\ & \cdot (q_0, *)\#\underset{S}{a}(h, *)\$(q_0, *)\#a(h \underset{S}{,} *)\$(q_0, *)\#\underset{S}{\overline{a}}(h, *)\$ \\ & \cdot \#\overline{\underset{(q_0,*)}{a}}(h, *)\$\#a\overline{\underset{(q_0,*)}{(h, *)}}\$\#\overline{(q_0 \underset{S}{,} *)}\$T, \end{aligned} \tag{18}$$

where

$$
\begin{aligned}
w = I[\#\underset{S}{(q_0,*)}\$][\#][a\underset{(q_0,*)}{(h,*)}\$][\#\underset{(q_0,*)}{a}(h,*)][\$] \\
\cdot [(q_0,*)][\#\underset{S}{a}(h,*)][\$][(q_0,*)][\#\underset{S}{\overline{a}}(h,*)][\$][\#\overline{\underset{(q_0,*)}{a}}(h,*)][\$] \\
\cdot [\#][a\overline{\underset{(q_0,*)}{(h,*)}}\$][\#\overline{\underset{S}{(q_0,*)}}\$]T.
\end{aligned}
\tag{19}
$$

Lemma 4.4 *The morphisms g and f are of bounded delay 2.*

Proof: We will prove this for g; the case of f is similar. The bounded delay condition of g can be violated in two ways: for two different letters r, s, either $g(r) = g(s)$ (which always violates it), or $g(r) < g(s)$ (which may or may not violate it). We will prove that the first case will not happen, and that if $g(r) < g(s)$, then there are no letters p, q such that $g(rpq)$ and $g(s)$ are comparable. This means that g is of bounded delay 2.

Assume that $g(r) = g(s)$. If $f_1([u]) = f_1([v])$, then also $g_1([u]) = g_1([v])$ by Table 1. If $g_1([u]) = g_1([v])$, then $u = v$, because $g_1([u]) = u$ and $g_1([v]) = v$. From this it follows that if $r, s \in A$, then $r = s$, and the same holds if $r, s \in \overline{A}$. If $r \in A \setminus \overline{A}$ and $s \in \overline{A} \setminus A$, then $g(r) \neq g(s)$, because $g(s)$ contains an element of $\overline{\Theta}$ and $g(r)$ does not. Finally, if $r = I$, then $s = I$, and if $r = T$, then $s = T$. It follows that g is injective on letters.

Assume that $g(r) < g(s)$. Now $g(s)$ contains $X \in \Theta \cup \overline{\Theta}$, but $g(r)$ does not contain it. If $g(rpq)$ and $g(s)$ are comparable, then $g(p)$ or $g(q)$ contains this X, but on a different position. This is impossible, because if X is a factor of $g(s)$ and $g(t)$, where $t \in \{p, q\}$, then its position in $g(s)$ and $g(t)$ is the same. This can be verified from Table 1. □

Theorem 4.5 *For any TM M we can construct two bounded delay 2 morphisms f, g and specify a letter I so that if M does not accept the empty word, then the set $\{w \in E(f, g) : I \leq w\}$ is empty, and if M does accept the empty word, then the set contains a unique minimal word.*

Proof: Follows from Lemma 4.2 and from Lemma 4.4. □

The undecidability result now follows with standard techniques.

Theorem 4.6 *The PCP for bounded delay 2 morphisms is undecidable.*

Proof: For any TM M we can construct the morphisms f and g of Theorem 4.5. If the modified PCP for bounded delay morphisms would be decidable, then it could be decided whether M accepts the empty word, but this is not possible.

The undecidability of the ordinary PCP for bounded delay morphisms follows easily from the modified version (see also [HK97]). For arbitrary bounded delay 2 morphisms $f, g : \Sigma_1^* \to \Sigma_2^*$ and letter $I \in \Sigma_1$ we will construct bounded delay 2 morphisms $f', g' : (\Sigma_1 \cup \{i, t\})^* \to (\Sigma_2 \cup \{i, t, X\})^*$ such that $\{w \in E(f, g) : I \leq w\}$ is empty if and only if $E(f', g')$ is empty (here i, t, X are new symbols).

Define morphisms l, r by $l(a) = Xa$ and $r(a) = aX$. Let $f'(x) = l(f(x))$ for $x \in \Sigma_1$, $f'(i) = il(f(I))$ and $f'(t) = Xt$. Let $g'(x) = r(g(x))$ for $x \in \Sigma_1$, $g'(i) = iXr(f(I))$ and $g'(t) = t$. Now f' and g' are of bounded delay 2. If $f'(w) = g'(w) \neq 1$, then $w = iutv$, where $u \in \Sigma_1^*$, and $f(Iu) = g(Iu)$. If $f(Iu) = g(Iu)$, then $f'(iut) = g'(iut)$. So $\{w \in E(f, g) : I \leq w\}$ is empty if and only if $E(f', g')$ is empty. □

5 Regularity of the Equality Set

To conclude our presentation we recall that the equality sets of bounded delay morphisms are regular. For the sake of completeness we present a short proof of this. More details can be found in [HK97]. Actually, even a stronger result is proved in [CK85].

Theorem 5.1 *Let $f, g : \Sigma \to \Gamma$ be morphisms of bounded delay. The language $E(f, g)$ is regular.*

Proof: Because f is of bounded delay, there is a number N such that if $v \in \Gamma^N$, then for at most one $a \in \Sigma$ there is a $w \in \Sigma^*$ such that $v \leq f(a)f(w)$. The same holds for g. We assume that N is a suitable bound for both morphisms f and g.

We construct a deterministic automaton accepting $E(f, g) \cup \{1\}$. Let $Q = (\Gamma^* \times \{1\}) \cup (\{1\} \times \Gamma^*)$ be the set of states and let

$$\delta((u, v), a) = \begin{cases} (u', 1) & \text{if } v = 1 \text{ and } uf(a) = g(a)u', \\ (1, v') & \text{if } v = 1 \text{ and } uf(a)v' = g(a), \\ (u', 1) & \text{if } u = 1 \text{ and } f(a) = vg(a)u', \\ (1, v') & \text{if } u = 1 \text{ and } f(a)v' = vg(a), \end{cases} \tag{20}$$

where $(u, v) \in Q$ and $a \in \Sigma$, be the (partial) transition function. So $\delta((u, v), a)$ is well defined, if $uf(a)$ and $vg(a)$ are comparable, and undefined otherwise.

Let $(u, v), (u', v') \in Q$ and $w \in \Sigma^*$. Now $\delta((u, v), w) = (u', v')$, if $uf(w)v' = vg(w)u'$, and $\delta((u, v), w)$ is undefined, if $uf(w)$ and $vg(w)$ are not comparable. In particular, $\delta((1, 1), w) = (1, 1)$ if and only if $f(w) = g(w)$. Thus the automaton $(Q, \Sigma, \delta, (1, 1), \{(1, 1)\})$, where $(1, 1)$ is the initial state and $\{(1, 1)\}$ is the set of final states, would accept $E(f, g) \cup \{1\}$, but Q is not finite. Restricting the automaton to the states occurring in chains of transitions of the form $(1, 1) \to q_1 \to \cdots \to q_n \to (1, 1)$ does not change the accepted language. We show that the set Q' of these states is finite, which proves the theorem.

Let $Q_1 = \{(u, v) \in Q' : |uv| < N\}$ and let Q_2 be the set of those states in $Q' \setminus Q_1$ to which there is a transition from Q_1. The sets Q_1 and Q_2 are finite. If $(u, v) \in Q' \setminus Q_1$, $a \in \Sigma$ and $\delta((u, v), a) \in Q'$, then $\delta((u, v), aw) = (1, 1)$ for some $w \in \Sigma^*$. This means that $uf(a)f(w) = vg(a)g(w)$. Thus either $u = 1$, $|v| \geq N$ and $v \leq f(a)f(w)$, or $v = 1$, $|u| \geq N$ and $u \leq g(a)g(w)$. In both cases a is uniquely determined because of the bounded delay property, so from each $q \in Q' \setminus Q_1$ there is a transition to only one state of Q'. Thus for each $q \in Q_2$ there is only one chain of the form $q \to q_1 \to \cdots \to q_n \in Q_1$, where $q_1, \ldots, q_{n-1} \notin Q_1$. Let Q_3 be the set of states in these chains. Because Q_2 is finite and each of these chains is finite, also Q_3 is finite. If $q \in Q' \setminus Q_1$, there is a chain of transitions from $(1, 1)$ to q, and another from q to $(1, 1)$. The former chain necessarily contains a state of Q_2. Thus $q \in Q_3$. This means that $Q' = Q_1 \cup Q_3$, so Q' is finite. □

To summarize, we have proved:

Theorem 5.2 *The equality language of bounded delay 2 morphisms is regular, but cannot be found algorithmically, and PCP for these morphisms is undecidable.*

We conclude by emphasizing that our Theorem 5.2 is not the strongest possible – K. Ruohonen proved in [Ruo85] a stronger result (for biprefix morphisms instead of bounded delay morphisms). However, our proof and constructions are much simpler, and can be viewed as “textbook proofs” of this important result.

References

[Ben73] Charles H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.

[CK85] Christian Choffrut and Juhani Karhumäki. Test sets for morphisms with bounded delay. *Discrete Appl. Math.*, 12(2):93–101, 1985.

[CK97] Christian Choffrut and Juhani Karhumäki. Combinatorics of words. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 329–438. Springer-Verlag, 1997.

[Cul79] Karel Culik, II. A purely homomorphic characterization of recursively enumerable sets. *J. ACM*, 26(2):345–350, 1979.

[ER78] Andrzej Ehrenfeucht and Grzegorz Rozenberg. Elementary homomorphisms and a solution of the D0L sequence equivalence problem. *Theoret. Comput. Sci.*, 7(2):169–183, 1978.

[Hai69] Leonard H. Haines. On free monoids partially ordered by embedding. *J. Combin. Theory*, 6:94–98, 1969.

[Hir03] Mika Hirvensalo. *Quantum Computing*. Springer-Verlag, second edition, 2003.

[HK97] Tero Harju and Juhani Karhumäki. Morphisms. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 439–510. Springer-Verlag, 1997.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[Kar09] Juhani Karhumäki. On the power of cooperating morphisms via reachability problems. *Internat. J. Found. Comput. Sci.*, 20(5):803–818, 2009.

[KS79] Juhani Karhumäki and Imre Simon. A note on elementary homomorphisms and the regularity of equality sets. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 9:16–24, 1979.

[Lec63] Yves Lecerf. Récursive insolubilité de l'équation générale de diagonalisation de deux monomorphismes de monoïdes libres $\varphi x = \psi x$. *C. R. Acad. Sci. Paris*, 257:2940–2943, 1963.

[Lot02] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

[Pos46] Emil Post. A variant of a recursively unsolvable problem. *Bull. Amer. Math. Soc.*, 52:264–268, 1946.

[Ruo85] Keijo Ruohonen. Reversible machines and Post's correspondence problem for biprefix morphisms. *Elektron. Informationsverarb. Kybernet.*, 21(12):579–595, 1985.