Available online at www.sciencedirect.com

Electronic Notes in Theoretical Computer Science 223 (2008) 15–27

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Reachability via Cooperating Morphisms

Juhani Karhumäki[1,2]

Department of Mathematics
University of Turku
Turku, Finland

Abstract

The goal of this presentation is to analyze the equality mechanism of cooperating morphisms of free monoids, and to point out that the reachability questions lead to the undecidability and easy characterizations of recursively enumerable languages. In particular, we aim to show, which subconstructions are needed in such results. Moreover, we recall that in some cases the undecidability of the reachability is achieved although the sets of all reachable objects are simple, or more formally, regular languages.

Keywords: morphisms, equality languages, recursively enumerable languages, reachability problems

---

1 Introduction

We consider *morphisms* of free monoids, that is, mappings $h : \Sigma^* \to \Delta^*$ satisfying $h(uv) = h(u) \cdot h(v)$, for all $u, v \in \Sigma^*$. By an *equality mechanism of two morphisms* h *and* g we mean a *selector* or a *filter*, which takes only those words w, such that $h(w)$ and $g(w)$ are in prefix relation, that is, there exists a word z, such that

$$h(w)z = g(w) \quad \text{or} \quad h(w) = g(w)z. \tag{1}$$

Such a filter can be viewed as a mapping

$$F_{h,g} : \Sigma^* \to \{0, 1\},$$

or as a mapping

$$FW_{h,g} : \Sigma^* \to \Delta^*,$$

---

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

[2] Email: `karhumak@utu.fi`

doi:10.1016/j.entcs.2008.12.028

where in the first case $F_{h,g}(w) = 1$, if and only if (1) holds true for some z, and $FW_{h,g}(w) = z$, if (1) holds true with the value z. There are two obvious observations:

(i) Whenever $F_{h,g}(wa)$, with $a \in \Sigma$, assumes value 1, so does $F_{h,g}(w)$;

(ii) Whenever $FW_{h,g}(wa)$, with $a \in \Sigma$, is defined, so is $FW_{h,g}(w)$.

Actually, in the second statement we can even write

$$FW_{h,g}(wa) \in \{\big(FW_{h,g}(w)g(a)\big)^{-1}h(a), h(a)^{-1}\big(FW_{h,g}(w)g(a)\big)\}$$

or

$$FW_{h,g}(wa) \in \{\big(FW_{h,g}(w)h(a)\big)^{-1}g(a), g(a)^{-1}\big(FW_{h,g}(w)h(a)\big)\}$$

depending on whether $FW_{h,g}(w) = z$ satisfies the first or the second condition in (1). Note also that the value of $FW_{h,g}(wa)$ in both of these cases is uniquely determined by relative lengths of $h(a)$ and $g(a)$.

The above mechanism can be viewed as a computation procedure, when starting from a fixed w satisfying "$h(w)$ and $g(w)$ are in prefix relation". Of course, such a process can be nondeterministic or deterministic, as well as finite or infinite. Hence, reachability questions come in a natural way into the game. We can ask whether such a computation is finite or reaches a particular situation or configuration. We are going to demonstrate that such definitionally natural and simple looking problems are extremely complicated.

More background material can be found, e.g., in handbook chapters [2] and [8].

2 Equality sets

The theory of equality languages fits well into the above formalism. The *equality set of two morphisms* $h, g : \Sigma^* \to \Delta^*$ is the set

$$E(h, g) = \{w \in \Sigma^* \mid h(w) = g(w)\}.$$

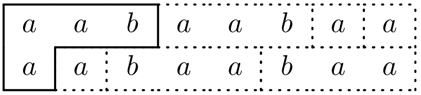
That is, $E(h, g)$ is the maximal set of words for which h and g are equal word by word. In terms of our reachability problems it constitutes of exactly those words w for which our filter $FW_{h,g}$ gives the value ε. Another variant of our reachability problem might ask whether our filter is passed by an infinite word.

Following three examples illustrate the above cases.

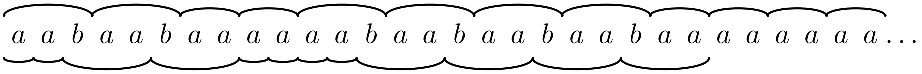
Example 2.1 For morphisms

$$h : \begin{array}{l} a \mapsto a \\ b \mapsto baa \end{array} \qquad g : \begin{array}{l} a \mapsto aab \\ b \mapsto a \end{array}$$

we have $E(h, g) = (aabb)^*$. Now, the above as a computation process is illustrated by the following figure



Example 2.2 If in the above example $g(b)$ is replaced by $g(b) = aa$, then no element of $E(h, g)$ (except ε) can be found, but the infinite word $\prod_{i=1}^{\infty} a^{2^i} b^{2^i}$ goes through the filter as follows:


a a b a a b a a a a a a b a a b a a b a a b a a a a a a a a . . .

Example 2.3 This example shows more clearly how the above mechanism can be viewed to compute something. Consider the morphisms:

$$s: \begin{array}{l} 1 \mapsto i \\ 2 \mapsto ab \\ 3 \mapsto \sharp \\ 4 \mapsto \sharp a \\ 5 \mapsto baba \\ 6 \mapsto \sharp baf \end{array} \qquad p: \begin{array}{l} 1 \mapsto iab\sharp \\ 2 \mapsto abab \\ 3 \mapsto \sharp \\ 4 \mapsto a\sharp \\ 5 \mapsto ba \\ 6 \mapsto f \end{array}.$$

Now, comparable images of letters are 1, 2, 3 and 5. Since $s(3) = \sharp = p(3)$, the word 3 is in $E(s, p)$, and nothing interesting comes out here. Neither give the initial values 2 or 4 anything interesting: they define the prefixes of 2^ω (and 4^ω) as filtered words. The fourth possibility, that is the letter 1, is more interesting. Indeed, for example,

$$v_3 = 1232232222455553355356$$

is in $E(s, p)$, since the word

$$w_3 = iab\sharp(ab)^2\sharp(ab)^4\sharp(ab)^8a\sharp(ba)^4\sharp(ba)^2\sharp baf$$

can be parsed with the equality mechanism, by s and p; that is,

$$s(v_3) = w_3 = p(v_3).$$

It is not difficult from above to conclude the general formula for the set of all *minimal* elements in $E(s, p)$, where minimal refers to the fact that no word is a prefix of another in the set. In particular, $E(s, p)$ is not a regular language, while s is a suffix set and p is a prefix set. Note also, that in the above construction the middle marker 4 is used to change the period ab to that of ba — and hence the counting becomes possible.

The equality sets determine exactly all solutions of instances of Post Correspondence Problem, which is probably the best known representative of combinatorial undecidable problems, cf. [10] and [8]. Hence, our point is not to emphasize the undecidability of our reachability problems as such, but rather to try to find basic and clear constructions needed in such proofs.

3 Basic reachability problem

We define an *individual reachability problem* for triples (α, h, g), where $h, g : \Sigma^* \to \Delta^*$ are morphisms and $\alpha \in \Sigma^+$, as the decision question, whether there exists a word w, such that

$$FW_{h,g}(\alpha w) \in \Delta^* \S \Delta^*,$$

where $\S$ is a special symbol in Δ. Our first result is

Theorem 3.1 *Individual reachability problem is undecidable.*

Proof. Our proof constitutes of straightforward simulation of the Halting Problem for Turing machines. We try to be selfcontained but brief in our presentation.

Assume that $\mathcal{M}$ is a Turing machine, with Q as the set of states and Γ as the total set of tape symbols. The transitions of the machine are of the form

$$(p, a) \to (q, b, L/R). \tag{2}$$

We assume that the machine is deterministic, 1-way and complete (meaning that there exists always the next move except when p above is the halting state q_h). As usual, configurations of $\mathcal{M}$ are words of the form

$$\alpha p \beta \quad \text{with } \alpha \in \Gamma^*, \beta \in \Gamma^+ \text{ and } p \in Q.$$

Each rule (2) changes a factor cpa in a configuration to the factor qcb or cbq depending on whether the third component in the right is L or R, respectively. In addition, the machine has a designated initial state q_0.

Now, the fundamental Halting Problem for Turing machines (of above restricted type) asks whether a given Turing machine $\mathcal{M}$ halts on a given input w. That is, whether the sequence of configurations

$$q_0 w = w_0, w_1, w_2, \ldots, \tag{3}$$

where w_{i+1} is obtained from w_i under the above rewriting determined by transitions of $\mathcal{M}$, is finite or infinite. In other words, whether $\mathcal{M}$ ever enters to the state q_h. This is typically the first example of undecidable problems.

Now, we conclude how our theorem follows from this. We have to define an instance (α, h, g), with $\alpha \in \Sigma^+$, $h, g : \Sigma^* \to \Delta^*$, such that there exists a word w such that

$$\left.\begin{array}{c} FW_{h,g}(\alpha w) \in \Delta^*\S\Delta^* \\ \\ \text{if and only if} \\ \\ \mathcal{M} \text{ halts on input w.} \end{array}\right\} \qquad (4)$$

In order to construct h and g we encode the computation into the form

$$Iw_0\sharp w_1\sharp w_2\sharp\cdots,$$

where I is so called initial marker and $\sharp$ is a separator. We choose $\alpha = i \in \Sigma$, $\S = q_h \in \Delta$ and set $h(i) = Iq_0w\sharp$ and $g(i) = I$, illustrated as

$$\overbrace{\underbrace{I}_{g(i)}\, q_0\, w\, \sharp}^{h(i)}$$

We assume that configurations $\alpha p\beta$ are actually of the form $\alpha p\bar\beta$, where $\bar\beta$ is a barred copy of β, or in fact in the spirit of Example 2.3, letters in α are written in the form $*a = a$ and in $\bar\beta$ in the form $b* = \bar b$. Now, we define

$$h(i_x) = x = g(i_x) \quad \text{for } x = \sharp, *a \text{ or } b*,$$

and further

$$g(i_t) = *cpa* \qquad \text{and} \qquad h(i_t) = new_t,$$

if t is the transition

$$(p, a) \to (q, b, L) \quad (\text{resp. } (q, b, R))$$

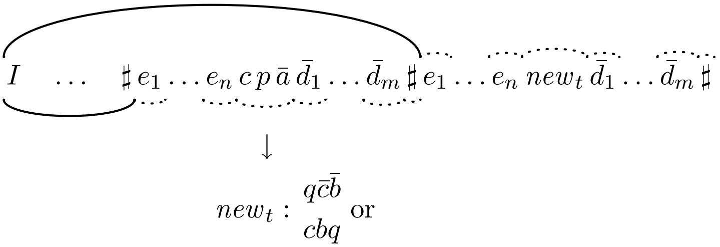
and

$$new_t = qc{*}b{*} \quad (\text{resp. } {*}c{*}bq).$$

It follows that assuming inductively that

$$\overbrace{\underbrace{I \quad \dots \quad \sharp}_{g(i_w)}\, e_1 \dots e_n\, c\, p\, \bar a\, \bar d_1 \dots \bar d_m\, \sharp}^{h(i_w)}$$

the only way to parse $g(i_w)^{-1}h(i_w)$, by images of g, is as illustrated below:


$I \;\ldots\; \sharp\, e_1 \ldots e_n\, c\, p\, \bar{a}\, \bar{d}_1 \ldots \bar{d}_m \sharp\, e_1 \ldots e_n\, new_t\, \bar{d}_1 \ldots \bar{d}_m \sharp$
$\downarrow$
$new_t :$ $q\bar{c}\bar{b}$ or cbq

Consequently, the restrictions on $\mathcal{M}$ immediately imply the equivalence (4), and hence also the proof of the theorem. □

4 PCP as a reachability problem

In the previous section we showed how computations of a Turing machine on a given input can be simulated by a pair of cooperating morphisms, and thus the individual reachability problem for morphisms is undecidable. Our construction is quite general and easily allows many variants of "reachability": We required that the reached configuration contains a special symbol. Equally we could require that "reachability" means that an infinite word passes our filter or that at the end the morphisms are not only in the prefix relation, but actually match to each other. The second variant corresponds to the famous Post Correspondence Problem, while the former to that for infinite words — or to be precise, to the modified versions, where first symbols in solutions are fixed.

Both of these, and many other situations, are easily shown undecidable, since the corresponding problems for Turing machines are so, and our approach allows simulations similar to those explained in the previous section. We prove here the case of the *Modified Post Correspondence Problem* asking to decide whether for two morphisms $h, g : \Sigma^* \to \Delta^*$ and a letter $i \in \Sigma$, the set

$$E_i(h, g) = \{w \in i\Sigma^* \mid h(w) = g(w)\}$$

is nonempty, cf. [8].

Theorem 4.1 *Modified Post Correspondence Problem is undecidable*

Proof. We use the notation of the proof of Theorem 3.1. The crucial idea is as follows: We extend a halting computation of a Turing machine $\mathcal{M}$ on w to the following generalized computation

$$Iw_0\sharp w_1\sharp \ldots \sharp \alpha a q_h \bar{b}\bar{\beta}\sharp \alpha a q_h \bar{\beta}\sharp \ldots \sharp \alpha a q_h \sharp \alpha a \bar{q}_h \sharp \alpha \bar{q}_h \sharp \ldots \sharp \bar{q}_h T.$$

Of course, this is not a computation of $\mathcal{M}$, but is described, as a computation, where, in addition, there are rewriting rules like

$$q_h\bar{b} \to q_h, \qquad q_h\sharp \to \bar{q}_h\sharp \qquad \text{and} \qquad a\bar{q}_h \to \bar{q}_h.$$

As we showed in the proof of Theorem 3.1 we can find morphisms $h, g : \Sigma^* \to \Delta^*$, such that

$$\overbrace{\underbrace{I\, w_0 \,\sharp \quad \ldots \quad \sharp}_{g(i_{w_n})} \,\alpha\, a\, q_h\, \bar{b}\, \bar{\beta}\, \sharp}^{h(i_{w_n})}$$

Now, since q_h has not appeared earlier, we can easily extend the domains of h and g, such that for some $i_{w_{n+1}}$ we have

$$\overbrace{\underbrace{I\, w_0 \,\sharp \quad \ldots \quad \sharp\, \alpha\, a\, q_h\, \bar{b}\, \bar{\beta}\, \sharp}_{g(i_{w_{n+1}})} \,\alpha\, a\, q_h\, \bar{\beta}\, \sharp}^{h(i_{w_{n+1}})}$$

Repeating the process and changing q_h to $\bar{q}_h$ (when we start to erase immediately before the state q_h) we finally succeed to extend h and g, such that for some word i_{w_m} we have the matching $h(i_{w_m}) = g(i_{w_m})$, proving the theorem. □

The above shows, we believe, that the Modified Post Correspondence Problem and it's undecidability is a very natural and clear example of reachability problems in our terminology.

To go from the Modified PCP to the ordinary PCP is not difficult, as is well known, see e.g. [8]. This is essentially done by the argument of shifting periods shown already in Example 2.3.

5 Reachability and language generation

In our formulation of reachability problems so far the initial word was part of the input as it is in the Halting Problem for Turing machines. On the other hand, Turing machines can be used as language generators by selecting those inputs, which allow a halting computation. The very same holds in our formulation of reachability problems for morphisms, and moreover, the power of the equality mechanism of two morphisms makes this extension quite straightforward.

This is the goal of this section. More concretely, we shall show how recursively enumerable languages can be obtained from equality languages, that is, via our reachability approach, by using in addition only simple language operations.

The contents of the next important result were proved simultaneously in slightly different forms in three different papers [4], [5] and [12].

Theorem 5.1 *For each recursively enumerable language L, there exist a regular language R and morphisms h, g and π, such that*

$$L = \pi\big(R \cap E(h, g)\big).$$

Proof. Let L be accepted by a Turing machine, say $\mathcal{M}$, of the type described in Section 3. We associate to each accepted word $w = a_1 \cdots a_n$ of $\mathcal{M}$ a generalized computation of $\mathcal{M}$ of the form:

$$Iq\sharp qa'_n\sharp \ldots \sharp qw'\sharp \underbrace{q_0\bar{w}}_{w_0} \sharp w_1 \sharp \ldots \sharp \alpha q_h \bar{\beta} \sharp \ldots \sharp \bar{q}_h T \tag{5}$$

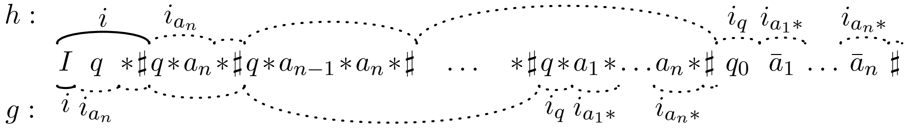
constituting of three parts: the prefix up to q_0, the factor from q_0 to the first occurrence $\alpha q_h \bar{\beta}$ and the suffix from there on.

The first part corresponds the generation of an input, the second part (accepting) computation of $\mathcal{M}$ on w, and the last one the erasing of the tape (and hence matching the cooperating morphisms). We shall show that the above words can be filtered by equality mechanism of two morphisms. Actually, by what we have already seen, it is enough to show that part one can be defined by our method, that is, that we can construct morphisms h and g for which the following holds for some word i_w

$$\overbrace{\underbrace{I\, q\, \sharp\, q\, a'_n\, \sharp \quad \ldots \quad \sharp\, q\, a'_1 \ldots a'_n\, \sharp}_{g(i_w)}\, q_0\, \bar{a}_1 \ldots \bar{a}_n\, \sharp}^{h(i_w)}$$

Moreover, the morphisms h and g have to be defined such, that they do uniformly this for all input words w, that is, that they can generate all potential input words. Hence, the process is inherently nondeterministic. This, however, is no problem, since the morphisms are allowed to be noninjective.

The above is realized as depicted below (where instead of primed letters a' we use factors $*a$; cf. the period shifting of Example 2.3):


h :
i
i_{a_n}
i_q
i_{a_1*}
i_{a_n*}
$I\ q\ *\sharp q*a_n*\sharp q*a_{n-1}*a_n*\sharp\ \ldots\ *\sharp q*a_1*\ldots a_n*\sharp\ q_0\ \bar{a}_1\ \ldots\ \bar{a}_n\ \sharp$
g :
i
i_{a_n}
i_q
i_{a_1*}
i_{a_n*}

What happens here is as follows. We start from a new state q. When g (which is behind) decodes q, it generates randomly an input letter a in the shifted form $*a$. These input letters are preserved under parsing by g. A crucial thing here is that in any point g is decoding q it can, besides above, decide that the whole input is already created and accordingly transform the q to the initial state q_0 of $\mathcal{M}$, and simultaneously change the shifted input $*a_1 \cdots *a_n$ to the actual input $\bar{w} = \bar{a}_1 \cdots \bar{a}_n$. This together with our earlier constructions guarantees that we can select by the equality mechanism exactly the encodings (5). But from this we can easily extract the input $w = a_1 \cdots a_n$. We only have to use different indices, namely i_{a*} (not used anywhere else), when decoding the occurrences of w just before q_0 by g, and specify the index (say t) encoding the final marker T. Then π can be defined such, that it erases everything else, but transforms the indices of the above word w to the actual letters of this word. To conclude, we set $R = (\mathrm{dom}(h) \setminus \{t\})^* t$. $\square$

The following remark follows directly from the above proof. Obviously, any equality set is a submonoid of a free monoid. Hence, it possesses the unique minimal generating set; let us refer to such a set as *minimal equality set*. Then, if equality sets are replaced by minimal equality sets, we can get rid of the intersection with a regular set. This indeed was the result of [4].

Theorem 5.2 *Each recursively enumerable set can be expressed as a morphic image of a minimal equality set.*

The above theorem is a splendid example of the power of morphisms of free monoids, by giving a purely morphic characterization of computability.

6 Complete twin shuffle languages as generators

In the previous section we showed that all recursively enumerable languages are generated by equality languages of morphisms of free monoids under a simple language-theoretic operations, namely closure under intersection with regular sets and morphic images. In this section we go a step further: We show that for this generation only very special equality sets are needed. This marvelous result was proved in [7].

We recall that the *shuffle* of two words u and v is the set of all words of the form $u_1 v_1 u_2 \cdots u_{n-1} v_{n-1} u_n$, with $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_{n-1}$. Let Σ_n be an n-letter alphabet and $\bar{\Sigma}_n$ its barred copy. We define, for $n \geq 1$, the *complete twin shuffle* TS_n over Σ_n as the language

$$TS_n = \{w \in (\Sigma_n \cup \bar{\Sigma}_n)^* \mid \overline{\pi_{\Sigma_n}(w)} = \pi_{\bar{\Sigma}_n}(w)\},$$

where π_{Σ_n} and $\pi_{\bar{\Sigma}_n}$ are projections of $(\Sigma_n \cup \bar{\Sigma}_n)^*$ into Σ_n^* and $\bar{\Sigma}_n^*$, respectively. It follows immediately that TS_n is an example of an equality language.

Now, the result of [7] can be stated as

Theorem 6.1 *For each recursively enumerable language L there exist an n, a morphism π and a regular language R, such that*

$$L = \pi(R \cap TS_n).$$

Proof. Again we rely on our earlier constructions. According to that, we can associate to each accepted input w of $\mathcal{M}$ the computation as shown in (5):

$$I \;\; \underset{\substack{\| \\ w_0}}{q} \; \sharp w_1 \sharp w_2 \ldots \sharp w_{n-1} \sharp \; \underset{\substack{\| \\ w_n}}{q_h} \, T, \tag{6}$$

where each w_i is a (generalized) configuration of $\mathcal{M}$, and, for all $i \geq 0$, w_{i-1} derives w_i in one step computation, that is, $w_i = \mathrm{Next}(w_{i-1})$. Recall that these generalized configurations are words of the form uqv, where u and v are words over the alphabet of $\mathcal{M}$ and q is a state. Consequently, the set of all potential configurations, say $\mathrm{Conf}(\mathcal{M})$, is a regular language. This holds also under the assumptions used in our earlier constructions, where in some places we used copies of letters. Further the words of the form (6) are also from a regular language. We denote this set by R'.

Now, we introduce crucial steps of our construction. For each $w_i \in \mathrm{Conf}(\mathcal{M})$, we define the indices

$$d_g(w_i) = g^{-1}(w_i) = a_1 \cdots a_n \tag{7}$$

and

$$d_h(w_i) = h^{-1}(w_i) = b_1 \cdots b_n \tag{8}$$

and the combined index

$$d(w_i) = a_1\bar{b}_1 \cdots a_n\bar{b}_n. \tag{9}$$

Consequently, d_g computes the index of w_i, with respect to g. Interestingly, this is unique, as our construction shows. Similarly, d_h computes the index of w_i under h, and this need not be unique, since the previous configuration in the computation of $\mathcal{M}$ need not be so.

Actually, two small clarifications of the above definitions are needed (and left to the reader): Namely the configurations w_{i-1} and w_i need not be exactly of the same length, and hence the indices in (7) and (8) might be of slightly different lengths. Another small thing is that the computations of the indices in (7) and (8) are in some cases dependent not only on w_i but also the surrounding markers.

Now, we are ready to conclude our proof. We associate to the word (6) the word

$$\bar{I}I d_g(q)\sharp d(w_1)\sharp\bar{\sharp} d(w_2)\ldots d(w_n)\sharp\bar{\sharp} d_h(q_h)\bar{\sharp}T\bar{T} \tag{10}$$

We choose R to be the set of words of the form (10) obtained from the words R' defined earlier. Clearly, R is regular, since we use in (9) only balanced shuffling. Implicitly the above fixes the size of Σ, that is n, needed here. It follows from above that

$$w \in R \cap TS_n$$

if and only if

$$d_g(w_{i-1}) = d_h(w_i)$$

and

$$d_g(w_{i+1}) = d_g\big(\mathrm{Next}(w_i)\big).$$

To summarize, R is used to consider sequences of correct configurations, and the complete twin shuffle to guarantee that the sequence of configurations is actually a computation of $\mathcal{M}$. Therefore, the result follows, if we choose π to pick up from (10) the input w. □

Our final step in sharpening the representation result of recursively enumerable languages is to eliminate the parameter n in Theorem 6.1. This requires the use of another operation, namely that of inverse morphic image. Indeed, let $\Sigma_n = \{a_1, \ldots, a_n\}$ and define $c : (\Sigma_n \cup \bar{\Sigma}_n)^* \to (\Sigma_2 \cup \bar{\Sigma}_2)^*$, with $\Sigma_2 = \{a, b\}$, by

$$c : \begin{array}{l} a_i \mapsto a^i b \\ \bar{a}_i \mapsto \bar{a}^i \bar{b} \end{array} \quad .$$

Then, obviously $TS_n = c^{-1}(TS_2)$, and we can formulate

Theorem 6.2 *For each recursively enumerable language L there exists a regular language R and morphisms π and c, such that*

$$L = \pi\big(R \cap c^{-1}(TS_2)\big).$$

We emphasize that all the operations used in the above theorem are *rational*, see [1]. Hence, there is a single, and at least definitionally simple, generator for the family of recursively enumerable languages under rational operations.

7 Regularity and undecidability

We conclude by pointing out one amazing result on equality languages originally proved in [11].

As we saw all recursively enumerable languages can be obtained from a fixed language, namely complete twin shuffle TS_2 over Σ_2, via rational operations. This means that TS_2 can be neither regular nor context-free. On the other hand, it is clearly recursive, or even context sensitive (since it is accepted by a deterministic Turing machine in linear space).

A question arises: Does there exists a class of morphisms for which the equality language would be regular? And if "yes", a further question would be the decidability of PCP for such a class of morphisms.

The first question was first answered affirmative in [5], where it was shown as a tool to resolve the famous D0L sequence equivalence problem, that the class of so-called *elementary* morphisms is such a class. Later it was noticed that it was not the elementariness, but so-called *bounded delay property*, which makes the equality sets regular, cf. [3]. We shall explain this in the moment.

After having this the second question became more concrete: Is the PCP for bounded delay morphisms decidable? The surprising negative answer to this was proved in [11], even in the case of prefix (or biprefix) morphisms.

To be more precise we recall that morphism $h : \Sigma^* \to \Delta^*$ possesses a bounded delay p, if the following holds:

$$h(au) < h(bv) \quad \& \quad |u| \geq p \qquad \Longrightarrow \qquad a = b.$$

Here the sign $<$ is used for the prefix relation. It follows that $p = 0$ corresponds to the fact that h is a prefix morphism, and, for any $p \geq 0$, bounded delay p implies that h is injective.

It is worth noting that (at least) morphism g in our earlier considerations does not have bounded delay for any p. And our results here shows that this is unavoidable.

A fundamental property of bounded delay morphisms related to the equality sets is as follows:

Lemma 7.1 *Let h and g have a bounded delay p. Then for any $u \in \mathrm{Pref}\,(E(h,g))$ satisfying*

$$h(u) = g(u)z \quad \textit{with } |z| \geq \max_{a\in\Sigma}\{|h(a)| \cdot p\}$$

there exists the unique letter b, such that $ub \in \mathrm{Pref}\,(E(h,g))$.

From this lemma it is straightforward to conclude

Theorem 7.2 *For any pair of bounded delay morphisms their equality language is regular.*

The above theorem motivates to study PCP for bounded delay morphisms. A striking answer was discovered:

Theorem 7.3 *The PCP for bounded delay morphisms is undecidable. Consequently, the regularity in Theorem 7.2 is nonconstructive.*

Above results deserve a few comments. First of all, Theorem 7.3 (even in the case of biprefix morphisms) was first proved in [11]. A simpler, more like a textbook type proof, was recently given in [9]. In the latter case p can be shown to be 2.

The proofs are based on the use of so-called *reversible* Turing machines and of two properties of those. Such machines are "globally injective" in the sense that for any configuration of the computation the previous one is uniquely defined. The two properties needed in the proof are that the halting problem for reversible Turing machines is undecidable, and that the computations of reversible Turing machine can be simulated, in the spirit of Section 3, by bounded delay morphisms

The result of this section clearly demonstrates, we believe, the power of the equality mechanism of two morphisms: The undecidability of PCP is not due to the fact that the set of all solutions is complicated.

Acknowledgement

The author is grateful to Markku Laine for helping in the preparations of this paper.

References

[1] Berstel, J., *Transductions and Context-Free Languages*, Teubner (1979).

[2] Choffrut, C., and J. Karhumäki, *Combinatorics of Words*, in G. Rozenberg and A. Salomaa (eds), Handbook of Formal Languages, Springer (1997).

[3] Choffrut, C., and J. Karhumäki, *Test sets for morphisms with bounded delay*, Discrete Appl. Math. **12** (1985) 93–101.

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

[5] Ehrenfeucht, A., and R. Rozenberg, *Elementary homomorphisms and a solution of the D0L sequence equivalence problem*, Theoret. Comput. Sci. **7** (1978), 169–183.

[6] Engelfriet, J., and G. Rozenberg, *Equality languages and fixed point languages*, Inform. Control **43** (1979), 20–49.

[7] Engelfriet, J., and G. Rozenberg, *Fixed Point Languages, Equality Languages, and Representation of Recursively Enumerable Languages* J. ACM **27** (1980), 499–518.

[8] Harju, T., and J. Karhumäki, *Morphisms*, in G. Rozenberg and A. Salomaa (eds), Handbook of Formal Languages, Springer (1997).

[9] Karhumäki, J., Saarela, A., *Noneffective regularity of equality languages and bounded delay morphisms*, manuscript, 12pp.

[10] Post, E., *A variant of a recursively unsolvable problem*, Bull. Amer. Math. Soc. **52** (1946), 264–268.

[11] Ruohonen, K., *Reversible machines and Post's correspondence problem for biprefix morphisms*, J. Inform. Process. Cybernet. (EIK) **21** (1985), 579–595.

[12] Salomaa, A., *Equality sets for homomorphisms of free monoids*, Acta Cybernetica **4** (1978), 127–139.