ELSEVIER

Theoretical Computer Science 255 (2001) 193–204

Theoretical Computer Science

www.elsevier.com/locate/tcs

Marked PCP is decidable

Vesa Halava[a,*], Mika Hirvensalo[b,a,1], Ronald de Wolf[c,d]

[a] *Turku Centre for Computer Science, Lemminkäisenkatu 14 A, 4th floor, FIN-20520, Turku, Finland*
[b] *Department of Mathematics, University of Turku, FIN-20014, Turku, Finland*
[c] *CWI, P.O. Box 94079, Amsterdam, Netherlands*
[d] *University of Amsterdam Netherlands*

Received September 1998; revised March 1999
Communicated by A. Salomaa

Abstract

We show that the *marked* version of the Post Correspondence Problem, where the words on a list are required to differ in the first letter, is decidable. On the other hand, we prove that the PCP remains undecidable if we only require the words to differ in the first *two* letters. Thus we locate the decidability/undecidability-boundary between marked and 2-marked PCP.

Keywords: Post Correspondence Problem; Marked morphism; Decidability

1. Introduction

The Post Correspondence Problem (PCP) [7] is one of the most useful undecidable problems, because of its simple, combinatorial description. Many other problems can easily be reduced to it, particularly problems in formal language theory. To define the general form of the problem we use a finite *source alphabet* $\Sigma = \{a_1, \ldots, a_n\}$, a finite *target alphabet* Δ and two morphisms $g, h: \Sigma^* \to \Delta^*$ ($g(ab) = g(a)g(b)$ and $h(ab) = h(a)h(b)$ whenever $a, b \in \Sigma^*$). An instance of the PCP is a four-tuple $I = (\Sigma, \Delta, g, h)$ and the PCP itself is the following decision problem:

Given $I = (\Sigma, \Delta, g, h)$, is there an $x \in \Sigma^+$ such that $g(x) = h(x)$?

* Corresponding author.
E-mail addresses: vehalava@cs.utu.fi (V. Halava), mikhirve@utu.fi (M. Hirvensalo), rdewolf@cwu.nl (R. de Wolf).

[1] Supported by the Academy of Finland under grant 44087.

0304-3975/01/$ - see front matter
PII: S0304-3975(99)00163-2

In other words, we have two lists of words $g(a_1),\ldots,g(a_n)$ and $h(a_1),\ldots,h(a_n)$ and we want to decide if there is a correspondence between them: are there $a_{i_1},\ldots,a_{i_k}\in\Sigma$ such that $g(a_{i_1})\ldots g(a_{i_k})=h(a_{i_1})\ldots h(a_{i_k})$? The word $x\in\Sigma^*$ such that $g(x)=h(x)$ is called a *solution* of I.

The general form of this problem is undecidable [7], the reason being that the two morphisms together can simulate the computation of a Turing machine on a specific input. Examining restricted versions of the PCP allows one to locate the boundary between decidability and undecidability. For instance PCP(1), where $n=1$, is trivially decidable and it turns out that also PCP(2) ($n=2$) is decidable [1]. On the other hand, PCP(7) remains undecidable [6] and presently the decidability status is open for source alphabet sizes $2<n<7$.

We may think about other kind of restrictions, too: For instance, the decidability of the problem is trivial if we restrict to solutions shorter than some fixed k, but this restricted form is **NP**-complete [2, p. 228]. If we restrict to g,h which have to be injective (g is injective if $x\neq y$ implies $g(x)\neq g(y)$), the problem still remains undecidable [4].

A stronger restriction than injectivity is to have g and h *marked*, which we formally define as follows. If z is a string, then $\mathrm{Pref}_k(z)$ stands for the prefix of length k of z ($\mathrm{Pref}_k(z)=z$ if $|z|\leqslant k$). A morphism g is *k-marked* if g is nonerasing ($g(a)$ is always nonempty) and $\mathrm{Pref}_k(g(a))\neq\mathrm{Pref}_k(g(b))$ whenever $a\neq b\in\Sigma$. An instance $I=(\Sigma,\Delta,g,h)$ of the PCP is *k-marked* if both g and h are k-marked, and k-marked PCP is the PCP decision problem restricted to k-marked instances. We will abbreviate 1-marked to *marked*. If I is marked then $g(a)$ and $g(b)$ start with a different letter whenever $a\neq b\in\Sigma$, which implies that $|\Sigma|\leqslant|\Delta|$, but without loss of generality we may even assume that $\Sigma\subseteq\Delta$. Markedness clearly implies injectivity (but k-markedness does not, in general): suppose g is marked and $x\neq y\in\Sigma^+$, let $x=zax'$ and $y=zby'$, a and b being the first letter where x and y differ. Because of markedness we have $g(a)\neq g(b)$, hence $g(x)=g(z)g(a)g(x')\neq g(z)g(b)g(y')=g(y)$, so g is injective. The converse does not hold. Consider for instance $\Sigma=\Delta=\{1,2\}$, $g(1)=11$, $g(2)=12$, then g is injective but not marked.

The proof of decidability of PCP(2) in [1] is based on a reduction from arbitrary instances of PCP(2) to marked instances of the *generalized* PCP(2), which is the following decision problem: Given morphisms $g,h:\Sigma^*\to\Delta^*$ and words $u_1,u_2,v_1,v_2\in\Delta^*$, is there a word $x\in\Sigma^+$ such that $u_1g(x)u_2=v_1h(x)v_2$? The authors then introduce a reduction procedure to convert an instance of generalized PCP to a (hopefully) simpler instance and eventually prove by extensive case analysis that the marked generalized PCP(2) is decidable. In particular the marked PCP(2) is decidable. Here we extend the reduction procedure of [1] and show that the marked PCP is decidable for *any* alphabet size. We will in fact show that the marked PCP is in **PSPACE** (the class of languages that can be recognized in space upper bounded by $p(N)$ for some polynomial p of the input size N).

As stated above, the PCP can be used for establishing the boundaries between decidability and undecidability. The main result of this paper is the decidability of the marked PCP. How much can we weaken the markedness condition before we lose decidability?

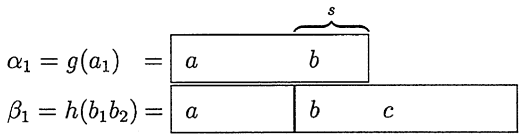

Fig. 1. Building a solution of marked PCP.

We will show in Section 7 that the 2-marked PCP is undecidable, thus locating the decidability/undecidability-boundary between 1-markedness and 2-markedness.

In another direction, we can weaken the markedness condition by only requiring g and h to be *prefix* morphisms (g is prefix if no $g(a_i)$ is a prefix of another $g(a_j)$) or even *biprefix* (g is biprefix if no $g(a_i)$ is a prefix or suffix of another $g(a_j)$). It turns out that the biprefix PCP is undecidable [8].[2]

2. Finding the decision procedure

A very obvious method to find solutions of an instance $I=(\Sigma,\Delta,g,h)$ of the marked PCP (if there are any) is to try to construct a solution x such that $g(x)=h(x)$ would begin with a particular $a\in\Delta$. Such a solution will be referred to as a *solution with label a* hereafter. Let us fix a label $a\in\Delta$. The attempt begins by choosing words $\alpha_0=g(a_1)$ and $\beta_0=h(b_1)$ that begin with a. If there are such words, they are unique because g and h are marked, and we check whether α_0 and β_0 are *comparable* (one of the words is a prefix of the other). Assume, for instance, that $\alpha_0=\beta_0 s$ for some $s\in\Delta^*$ which we call an *overflow of g* (overflow of h is defined analogously). Because h is marked, there is at most one $b_2\in\Sigma$ such that $h(b_2)$ begins with the initial letter b of s. If $h(b_2)$ is comparable with s, we define $\alpha_1=\alpha_0$ and $\beta_1=\beta_0h(b_2)$ (see Fig. 1).

In general, the procedure can be described by means of a sequence (α_i,β_i), where (α_0,β_0) is defined as above and

$$(\alpha_{i+1},\beta_{i+1})=\begin{cases}(\alpha_i,\beta_ih(c)), & \text{if } |\alpha_i|>|\beta_i| \text{ and } \alpha_i \text{ and } \beta_ih(c)\\ & \text{are comparable for some } c\in\Sigma.\\ (\alpha_ig(c),\beta_i), & \text{if } |\alpha_i|<|\beta_i| \text{ and } \alpha_ig(c) \text{ and } \beta_i\\ & \text{are comparable for some } c\in\Sigma.\end{cases}$$

Otherwise $(\alpha_{i+1},\beta_{i+1})$ remains undefined. Because g and h are marked, each (α_i,β_i) is unique, if defined. The process continues until one of the following cases occur:

1. Blocking case

If $|\alpha_i|>|\beta_i|$ (resp. $|\alpha_i|<|\beta_i|$) but α_i and $\beta_ih(c)$ (resp. $\alpha_ig(c)$ and β_i) are not comparable for any $c\in\Sigma$, we call the case *blocking* (Fig. 2).

[2] Clearly, a marked morphism is prefix. Both marked and biprefix PCP are special cases of the injective PCP, but the 2-marked PCP is not. See also at the end of Section 7.

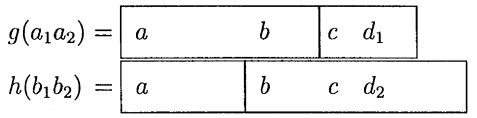

Fig. 2. Blocking case, $d_1 \neq d_2$.

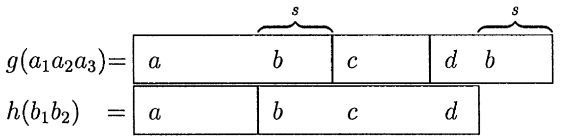

Fig. 3. Periodic case.

2. Periodic case

If an overflow of g or h is seen twice, the case is called *periodic*. Because the continuation is always unique, the process would cycle forever (Fig. 3).

3. Terminating case

If $\alpha_i = \beta_i$ for some i, we say that the case is *terminating* (Fig. 4).

If the case is not blocking or terminating, then it is periodic, since the overflows of g (resp. h) are proper suffixes of words $g(a_1), \ldots, g(a_n)$ (resp. $h(a_1), \ldots, h(a_n)$) and there are only finitely many such suffixes. The blocking and periodic cases are easy to handle: solutions (with label a) do not exist. On the other hand, the terminating case seems to be essentially more complicated: We just obtain words u and v that satisfy $\alpha_i = g(u) = h(v) = \beta_i$ and $|g(u)| = |h(v)|$ is minimal. Later we shall call such words u and v *blocks* with label a or *a-blocks*. Noncomparability of u and v clearly implies that solutions with label a do not exist. On the other hand, if $u = vw$ for some word $w \in \Sigma^*$, then either w is empty and a solution has been found, or w begins with some letter b. In the latter case we can continue the search by defining $(\alpha_{i+1}, \beta_{i+1}) = (\alpha_i, \beta_i h(b))$.

For example, Fig. 4 can not represent any solution, since the word $u = a_1a_2a_3$ is longer than $v = b_1b_2$. If $a_1a_2 = b_1b_2$, we just complete the image of h by adding $h(a_3)$ and continue the procedure. But the procedure may again end up in the terminating case, still not proving nor refuting the existence of a solution with label a. At this point, there is no a priori knowledge on how long the process should run until we can decide if there is a solution with label a or not.

On the other hand, if (Σ, Δ, g, h) has a solution, then it can evidently be represented as

$$g(u_1)g(u_2)\ldots g(u_k) = h(v_1)h(v_2)\ldots h(v_k),$$

where each (u_i, v_i) is a pair of blocks labelled with some $a_i \in \Delta$. Notice also that if (u_i, v_i) and (u_j, v_j) are pairs of blocks with labels $a_i \neq a_j$, then u_i and u_j (and also

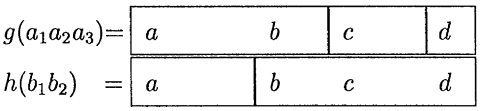

Fig. 4. Terminating case.

v_i and v_j) begin with a different letter, because morphisms g and h are marked. This means that the all the block pairs $(u_1, v_1), \ldots, (u_m, v_m)$ can be written as two lists of words that satisfy the markedness condition. To find a solution we should then find a sequence of indices such that $u_{i_i} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$, which means that *we are left with another instance of the marked PCP.* This leads to the approach of the next section.

3. Reduction to simpler instances

The aim is to reduce an instance $I = (\Sigma, \Delta, g, h)$ of the marked PCP to a somewhat easier instance $I' = (\Sigma', \Delta, g', h')$ of the same problem in a way that preserves *equivalence*: I has a solution if and only if I' does. To do this, we check for each label $a \in \Delta$ whether the procedure in Section 2 is terminating or not. Because Σ can be renamed, we can assume that letters $a_1, \ldots, a_m \in \Sigma \subseteq \Delta$ are exactly the labels that lead to the terminating case giving $(u_1, v_1), \ldots, (u_m, v_m)$ as the corresponding block pairs (of course, there cannot be more than $n = |\Sigma|$ labels leading to the terminating case). We now define the *reduced instance* $I' = (\Sigma', \Delta, g', h')$ by $\Sigma' = \{a_1, \ldots, a_m\} \subseteq \Sigma$ and $g'(a_i) = u_i$ and $h'(a_i) = v_i$ for each $a_i \in \Sigma'$. It is worth noticing that by construction, the concatenated morphisms gg' and hh' are identical: $g(g'(a_i)) = g(u_i) = h(v_i) = h(h'(a_i))$ for any letter $a_i \in \Sigma'$, so $gg'(w) = hh'(w)$ for each $w \in \Sigma'^*$ also. The crucial property of the reduction from I to I' is that the equivalence is preserved:

Lemma 1. *Let I' be the reduction of I. Then I and I' are equivalent.*

Proof. Assume first that I has a solution $g(x) = h(x)$ beginning with a_1. Let u_1 and v_1 be the prefixes of x such that the word $g(u_1) = h(v_1)$ has *minimal* length (u_1 and v_1 are clearly unique). This implies that (u_1, v_1) is the pair of a_1-blocks. Let $x = u_1 s_1 = v_1 t_1$. If $u_1 \neq x$, then also $v_1 \neq x$ and $g(s_1) = h(t_1)$, both words beginning with some letter a_2. Now there are unique prefixes u_2 and v_2 of s_1 and t_1, respectively, such that $g(u_2) = h(v_2)$ has minimal length. This again means that u_2 and v_2 are the a_2-blocks. Continuing in this way we can reveal two factorizations of $x = u_1 u_2 \ldots u_k = v_1 v_2 \ldots v_k$ such that (u_i, v_i) is the pair of a_i-blocks. By definitions of g' and h', $g'(a_1 \ldots a_k) = u_1 \ldots u_k = v_1 \ldots v_k = h'(a_1 \ldots a_k)$.

On the other hand, if I' has a solution x', then $x = g'(x') = h'(x')$ is a solution of I, since gg' and hh' are identical: $g(x) = gg'(x') = hh'(x') = h(x)$. □

If we could prove that I' is somehow simpler than I, then we could repeat the procedure, reduce to simpler and simpler equivalent instances I'', I''',..., and (hopefully) finally decide I. I' can be simpler than I in the sense that $|\Sigma'|<|\Sigma|$ $(m<n)$ and if the reduction eventually leads to alphabet size 1, we can trivially decide I. But it turns out that there are instances that do not lead to $|\Sigma|=1$ (see [4]), hence we need another way to measure how complex an instance I is.

4. Suffix complexity

For an instance $I=(\Sigma,\Delta,g,h)$ of marked PCP we define, analogously to [1], the *suffix complexity*:

$$\sigma(I) = \left|\bigcup_{a\in\Sigma}\{x \mid x \text{ is a proper suffix of } g(a)\}\right| + \left|\bigcup_{a\in\Sigma}\{x \mid x \text{ is a proper suffix of } h(a)\}\right|$$

and demonstrate that the reduction from I to I' cannot increase the suffix complexity. The intuitive idea behind the proof is that the reduction is based on building the blocks that become the images of the reduced morphisms. But a proper suffix of some $g'(a')$ is built because a proper suffix of some $h(a)$ is seen, so words $g'(a')$ cannot have more proper suffixes altogether than words $h(a)$ do. Similarly, words $h'(a')$ have at most equally many proper suffixes as words $g(a)$.

Lemma 2. *Let I' be the reduction of I. Then $\sigma(I')\leqslant\sigma(I)$.*

Proof. Define the following four sets:

$$G = \bigcup_{a\in\Sigma}\{x \mid x \text{ is a proper suffix of } g(a)\},$$

$$G' = \bigcup_{a\in\Sigma'}\{x \mid x \text{ is a proper suffix of } g'(a)\},$$

$$H = \bigcup_{a\in\Sigma}\{x \mid x \text{ is a proper suffix of } h(a)\},$$

$$H' = \bigcup_{a\in\Sigma'}\{x \mid x \text{ is a proper suffix of } h'(a)\}.$$

We will define an injective function $p: G'\to H$. Let $u=x_r\ldots x_c\in G'$ be a proper suffix of some $g'(a_i)=u_i=x_1\ldots x_c$. Let s be the shortest element of H that is comparable with $g(u)$ and appears as an overflow of h in a terminating case of the procedure of Section 2. At least one overflow comparable with $g(u)$ exists, because u_i itself is a a_i-block generated by the procedure and so x_r has been introduced because of seeing a proper suffix of some $h(y_t)$ comparable with $g(x_r)$ (see Fig. 5). Furthermore, the shortest such overflow s is unique because it is not only comparable with $g(x_r)$ but also with $g(x_rx_{r+1}\ldots x_c)$. Define p as $p(u)=s$.

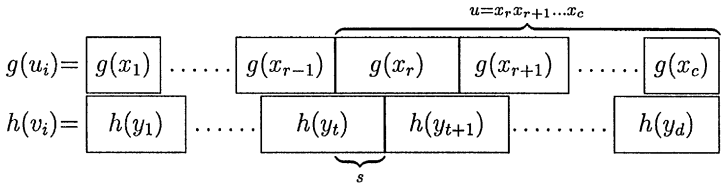

Fig. 5. The suffix s corresponding to u.

The injectivity of p is seen as follows: If $u = x_r x_{r+1} \dots x_c$ and $u' = x'_r x'_{r+1} \dots x'_d$ are elements of G' and $s = p(u) = p(u')$, then s is the shortest overflow of h due to which x_r and x'_r are introduced. Because of markedness, any overflow completely determines how the procedure continues, eventually giving $u = u'$. Thus p is injective, which implies $|G'| \leqslant |H|$.

Similarly we can define an injective function from H' to G, which proves $|H'| \leqslant |G|$. It now follows that $\sigma(I') = |G'| + |H'| \leqslant |G| + |H| = \sigma(I)$. □

If the consecutive reductions do not lead to source alphabet of size 1, we may hope that they eventually lead to an instance with $\sigma = 0$. Such an instance is clearly decidable, because then all words would have length 1. Unfortunately there are also instances that never reach $|\Sigma| = 1$ or $\sigma = 0$ (see [4]), but now we have a limitation on the number of distinct instances in the reduction procedure:

Lemma 3. *Let $\Sigma = \{a_1, \dots, a_m\} \subseteq \Delta$ be finite alphabets and z be a positive natural number. There exist only finitely many distinct instances $I = (\Sigma, \Delta, g, h)$ of the PCP that satisfy $\sigma(I) \leqslant z$.*

Proof. Recall first that there is nothing essential in Σ but the cardinality: An instance $I = (\Sigma, \Delta, g, h)$ is completely specified by giving the $2m$ words $g(a_1), \dots, g(a_m)$, $h(a_1), \dots, h(a_m) \in \Delta^+$. Note that if one of those words has length $> z + 1$, then this word has more than z proper suffixes and $\sigma(I) > z$. Accordingly, each of the $2m$ words can have length at most $z + 1$, so there are at most $(|\Delta| + 1)^{2m(z+1)}$ different I that satisfy $\sigma(I) \leqslant z$. □

It is now easy to see that if the sequence of reductions does not reach an I_j with alphabet of size 1 or $\sigma(I_j) = 0$, then the process starts to cycle: Assume that there exist k, m and z such that all I_i in the infinite sequence I_k, I_{k+1}, $I_{k+2}, \dots$ have source alphabet of size m and $\sigma(I_i) = z$. Now this sequence will repeat itself after a while, for otherwise there would be infinitely many distinct instances with the same alphabet and σ-value, contradicting Lemma 3.

5. Marked PCP is decidable

The decision procedure for the marked PCP is based on making equivalence-preserving reductions $I_0, I_1, I_2, \dots$ beginning with $I_0 = I = (\Sigma, \Delta, g, h)$ until one of the following

cases occur:
(1) Sequence reaches an I_j with $|\Sigma_j| = 1$.
(2) Sequence reaches an I_j with $\sigma(I_j) = 0$.
(3) Sequence starts cycling.
As seen before, cases (1) and (2) can be solved easily. Finally we will show that the instances leading to a cycle are easily solvable:

Lemma 4. *Let I be an instance of the marked PCP that starts a cycle (i.e. starting the reduction process with I eventually gives I again). Then I has a solution if and only if I has a solution of length* 1.

Proof. Assume that $I_0 = I$ eventually appears again:

$$I_0 \to I_1 \to \cdots \to I_{r-1} \to I_r = I_0,$$

where $I_i = (\Sigma, \Delta, g_i, h_i)$. By the proof of Lemma 1, for every solution x_i to some I_i, there is a solution x_{i+1} to I_{i+1} such that $x_i = g_{i+1}(x_{i+1}) = h_{i+1}(x_{i+1})$. Suppose x_0 is a solution to I_0 of minimal length. Applying the relation between two consecutive solutions, we find out inductively that there is some solution x_r to I_r such that

$$x_0 = g_1(x_1) = g_1g_2(x_2) = \cdots = g_1g_2 \ldots g_r(x_r),$$
$$x_0 = h_1(x_1) = h_1h_2(x_2) = \cdots = h_1h_2 \ldots h_r(x_r).$$

Since the g_i and h_i cannot be length-decreasing, we have $|x_0| \geqslant |x_r|$. But x_0 was chosen to be a minimal-length solution to I_0 and x_r is also a solution to $I_r = I_0$, hence $|x_0| = |x_r|$. This implies that $g_0(= g_r)$ and $h_0(= h_r)$ map the letters occurring in x_r to letters. But then the first letter of x_r is already a solution, hence $|x_0| = |x_r| = 1$. Thus I_0 has a solution if and only if I_0 has a 1-letter solution (i.e., there is an $a \in \Sigma_0$ such that $g_0(a) = h_0(a)$). □

Note that to decide if the reduction process has reached a cycle, we do not need to remember all the instances seen before, but it suffices to count *how many* instances with a fixed $m = |\Sigma|$ and suffix complexity z have been seen so far: If the counter exceeds $(|\Delta| + 1)^{2m(z+1)}$, then some instance has certainly occurred twice and the process is cycling. Below we summarize this analysis in an algorithm and a theorem:

Decision procedure for marked PCP.
(1) Set $c = 0$, $i = 0$, $I_0 = I$.
(2) Set $i = i + 1$.
(3) Reduce I_{i-1} to I_i in the way stated above.
(4) If I_i has source alphabet of size 1 or $\sigma = 0$, then decide I_i, print the outcome and terminate.
(5) If I_i is simpler than I_{i-1} (smaller source alphabet or σ) then set $c = 0$ and goto 2.

(6) If $c > (|\Delta| + 1)^{2m(z+1)}$, where m is the current alphabet size and z the suffix complexity, then there is a cycle and we can decide I_i by checking if it has a 1-letter solution, print the outcome and terminate;
else set $c = c + 1$ and goto 2.

Theorem 5. *Marked PCP is decidable.*

6. Complexity analysis

To end the decidability-part we analyze the complexity of the algorithm. Each reduction step can be done in linear space, if we ignore the space needed to print the outcome (i.e., the next instance). Namely, let N_i be the size of some instance I_i (i.e., the number of bits needed to describe the instance). The blocks are found by running the procedure of Section 2 for each label $a \in \Delta$. To decide if the procedure is terminating, we need only to remember the current overflow (requires $O(N_i)$ bits) and how many suffixes has been seen so far (if the counter exceeds the number of the suffixes, we know the procedure is in cycle). Since there are only $2n = O(N_i)$ different $g(a_i)$ and $h(a_i)$, there are only $O(N_i^2)$ different suffixes, hence $O(\log\ N_i)$ bits are enough for the counter. But how large can the reduced instances grow? If there would be some word $g(a_i)$ of an instance I_i longer than $\sigma(I_0) + 1$, then $\sigma(I_i) > \sigma(I_0)$ contradicting Lemma 2. Therefore $N_i = O(2n(\sigma(I_0) + 1)) = O(N^3)$ for any instance I_i and the space bound $O(N^3)$ bound for any reduction step follows.

In the decision procedure, the counter c runs up to $(|\Delta| + 1)^{(z+1)2m}$ and remembering that $m = O(N)$, $|\Delta| = O(N)$ and $z = O(N^2)$ we see that no more than $O(N^3 \log N)$ bits are needed to sustain the counter. Therefore, marked PCP is in **PSPACE**. The space bound $O(N^3 \log N)$ also implies a time bound $2^{O(N^3 \log N)}$.

7. 2-Marked PCP is undecidable

Here we will show that if we weaken the condition of markedness, by only requiring the morphisms to be 2-marked, then the PCP becomes undecidable again.

Consider the following semigroup $S_7 = \Gamma/R$ with set of 5 generators $\Gamma = \{a, b, c, d, e\}$ and 7 relations:

$$S_7 = \langle a, b, c, d, e \mid R \rangle$$

$$R = \{ac = ca, ad = da, bc = cb, bd = db, eca = ce, edb = de, cca = ccae\}.$$

Tzeitin [10] (see also [3, p. 445]) proved that the following problem for this semigroup is undecidable:

Given $u, v \in \Gamma^+$, is $u = v \in S_7$?

Table 1
Definition of g and h

	B	E	#	$\underline{\#}$	a	…	e	$\underline{a}$	…	$\underline{e}$	$[s=t]$	$[\underline{s}=\underline{t}]$
g	$Bu\#$	E	$\underline{\#}$	#	$\underline{a}$	…	$\underline{e}$	a	…	e	$\underline{t}$	s
h	B	$\underline{\#}vE$	#	$\underline{\#}$	a	…	e	$\underline{a}$	…	$\underline{e}$	s	$\underline{t}$

Note that the set of 7 left-hand-sides of R is 2-marked, and similarly for the set of 7 right-hand-sides of R. We will reduce this problem to the 2-marked PCP. We use a slight modification of the standard reduction, involving an alphabet with some underlined letters in order to ensure 2-markedness.

Define the source alphabet as

$$\Sigma = \Gamma \cup \underline{\Gamma} \cup \{B, E, \#, \underline{\#}, r_1, r_2, \ldots, r_7, \underline{r_1}, \underline{r_2}, \ldots, \underline{r_7}\},$$

where $\underline{\Gamma} = \{\underline{a}, \underline{b}, \underline{c}, \underline{d}, \underline{e}\}$, and $r_1, \ldots, r_7$ are the 7 relations in R and $\underline{r_1}, \ldots, \underline{r_7}$ are their underlined versions (considered as single letters), so $r_1 = [ac = ca]$, $\underline{r_1} = [\underline{ac} = \underline{ca}]$ etc. Define the target alphabet as

$$\Delta = \Gamma \cup \underline{\Gamma} \cup \{B, E, \#, \underline{\#}\}.$$

B and E will mark the beginning and end of expressions, respectively, and # and $\underline{\#}$ will act as separators. Given $u, v \in \Gamma^+$, g and h are defined by Table 1.

Note that the constructed instance $I = (\Sigma, \Delta, g, h)$ is an instance of the 2-marked PCP. The following lemma shows that the reduction preserves equivalence with Tzeitin's problem:

Lemma 6. *Let u, v, I be as above. Then $u = v \in S_7$ if and only if I has a solution.*

Proof. Suppose first that $u = v \in S_7$. Then there is a sequence $u = u_1 \to u_2 \to \cdots \to u_k = v$, where $u_i = u'su''$ and $u_{i+1} = u'tu''$, and $s = t \in R$ or $t = s \in R$. We construct a solution to I by induction on k.

If $k = 1$, then $u = v \in \Gamma^+$. Now $x = Bu\#\underline{u}E$ is a solution to I.

Now let $I' = (\Sigma, \Delta, g', h')$ be the instance of the 2-marked PCP corresponding to $u = u_{k-1} \in S_7$. By the induction hypothesis we can assume that I' has a minimal-length solution x'. It is easy to see that every solution must begin with B and end with E, so $x' = ByE$, and therefore $g'(By) = w\underline{\#}u_{k-1}$ and $h'(By) = w$ for some w. Note that since I and I' only differ in the assignment $h(E)$ and $h'(E)$, and E cannot occur in y (because x' is minimal), we also have $g(By) = w\underline{\#}u_{k-1}$ and $h(By) = w$. We distinguish two cases. Firstly, $u_{k-1} = u'su''$ and $v = u_k = u'tu''$, where $r = [s = t]$ is one of the 7 relations. Then it is easily verified that $x = By\underline{\#}u'ru''\#\underline{u'tu''}E$ is a solution to I. Secondly, if $u_{k-1} = u'tu''$ and $v = u_k = u'su''$, then $x = By\underline{\#}u'tu''\#\underline{u'ru''}E$ is a solution. This completes the induction step.

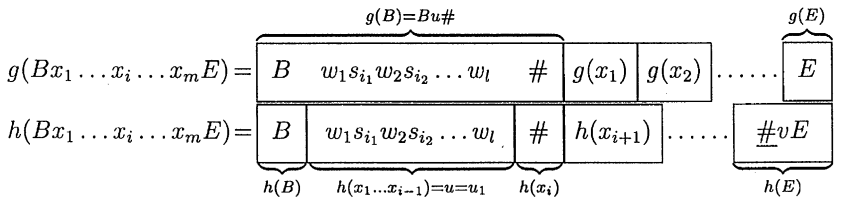

Fig. 6. Picture leading to $u = v$.

For the other direction, suppose I has a minimal-length solution x. This x must be of the form $Bx_1x_2 \ldots x_mE$, where $x_i \in \Sigma$, so $g(Bx_1 \ldots x_mE) = Bu\#g(x_1 \ldots x_m)E = Bh(x_1 \ldots x_m)\underline{\#}vE = h(Bx_1 \ldots x_mE)$. Ignoring the underlining, $g(x) = h(x)$ must be of the form $Bu_1\#u_2\# \ldots \#u_{k-1}\#u_kE$, where $u_i \in \Gamma^*$, $u_1 = u$ and $u_k = v$. We will show that $u_i = u_{i+1} \in S_7$ for every $1 \leqslant i \leqslant k-1$, from which $u = v \in S_7$ follows.

Since $Bu\#g(x_1 \ldots x_m)E = Bh(x_1 \ldots x_m)\underline{\#}vE$, # must occur in $h(x_1 \ldots x_m)$, so there is the smallest i such that $x_i = \#$, and hence $u = h(x_1 \ldots x_{i-1})$. Since there is no underlining in u, it follows that $x_1, \ldots, x_{i-1}$ must have been chosen from $a, \ldots, e, r_1, \ldots, r_7$. Let $x_1 \ldots x_{i-1} = w_1r_{i_1}w_2r_{i_2} \ldots w_l$, with $w_i \in \Gamma^*$ and $r_i = [s_i = t_i] \in \{r_1, \ldots, r_7\}$. Then $u = h(w_1r_{i_1}w_2r_{i_2} \ldots w_l) = w_1s_{i_1}w_2s_{i_2} \ldots w_l$. See Fig. 6 for illustration.

Note that $g(x_1 \ldots x_{i-1}) = g(w_1r_{i_1}w_2r_{i_2} \ldots w_l) = \underline{w_1t_{i_1}w_2t_{i_2} \ldots w_l}$. But now, since we must have $g(x_1 \ldots x_mE) = h(x_{i+1} \ldots x_mE)$, there must be the smallest index $j > i$ such that $x_j \in \{\#, \underline{\#}\}$ and $h(x_{i+1} \ldots x_{j-1}) = g(x_1 \ldots x_{i-1}) = \underline{w_1t_{i_1}w_2t_{i_2} \ldots w_l}$. The latter string (without underlining) is u_2. Note that $u_1 = u_2 \in S_7$, because $u_1 (= u)$ and u_2 only differ by u_2 having t_i where u_1 has s_i.

Continuing this reasoning, we can show that for every two words $u_i, u_{i+1} \in \Gamma^*$ occurring in $g(x) = h(x)$ separated by #, ignoring underlining, we must have $u_i = u_{i+1} \in S_7$ (some of the words u_i and u_{i+1} may actually already be equal in Σ^+). Hence u and v are equal in S_7, since $g(x)$ starts with $u_1 = u$ and ends with $u_k = v$. □

Together with Tzeitin's result, the above lemma implies:

Theorem 7. 2-*Marked PCP is undecidable.*

To end this section, we emphasize that 2-marked PCP is not a special case of the injective PCP. For example, the morphism defined by $g(1) = 23$, $g(2) = 2$, $g(3) = 3$ is 2-marked but not injective. We can combine k-markedness and injectivity by calling a morphism g *strongly k-marked* if g is both k-marked and prefix (i.e., no $g(a_i)$ is a prefix of another $g(a_j)$). This clearly implies injectivity. It follows from a construction of Ruohonen [8] that the strongly 5-marked PCP is undecidable: the biprefix instances of PCP constructed there to show undecidability of the biprefix PCP are also 5-marked. Decidability of the strongly k-marked PCP for $1 < k < 5$ is still open.

8. Conclusion and future work

We can investigate the boundary between decidability and undecidability by examining which restrictions on the Post Correspondence Problem render the problem decidable. We have shown here that restricting the PCP to *marked* morphisms gives us decidability. On the other hand, the 2-marked PCP is still undecidable.

The following questions remain open:

- Is polynomial space the best we can do when deciding the marked PCP or is the problem solvable even in polynomial time?
- What about decidability of the strongly k-marked PCP for $1 < k < 5$?
- What about decidability of the marked *generalized* PCP [1, 3]?
- The decidability status of the PCP with *elementary* morphisms [9, pp. 72–77] is still open. A morphism g is elementary if it cannot be written as a composition $g_2 g_1$ via a smaller alphabet. The marked PCP is a subcase of the elementary PCP which we have shown here to be decidable. Can our results help to settle the decidability status of the elementary PCP?

Acknowledgements

We thank Tero Harju, Juhani Karhumäki, John Tromp and Harry Buhrman for helpful comments. The second author would like to thank the CWI for its hospitality during the summer of 1998, when part of this work was done.

References

[1] A. Ehrenfeucht, J. Karhumäki, G. Rozenberg, The (generalized) Post correspondence problem with lists consisting of two words is decidable, Theoret. Comput. Sci. 21(2) (1982) 119–144.
[2] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[3] T. Harju, J. Karhumäki, D. Krob, Remarks on generalized Post correspondence problem, Proceedings of 13th STACS, Lecture Notes in Computer Science, Vol. 1046, Springer, Berlin, 1996, pp. 39–48.
[4] T. Harju, J. Karhumäki, Morphisms, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 1, Springer, Berlin, 1997, pp. 439–510.
[5] Y. Lecerf, Récursive insolubilité de l'équation générale de diagonalisation de deux monomorphisms de monoïdes libres $\phi x = \Psi x$. Comptes Rendus Acad. Sci. Paris 257 (1963) 2940–2943.
[6] Y. Matiyasevich, G., Sénizergues, Decision problems for semi-Thue systems with a few rules, Proceedings of the 11th IEEE Symposium on Logic in Computer Science, 1996, pp. 523–531.
[7] E.L. Post, A variant of a recursively unsolvable problem, Bull. Amer. Math. Soc. 52 (1946) 264–268.
[8] K. Ruohonen, Reversible machines and Post's correspondence problem for biprefix morphisms, J. Inform. Process. Cybernet. (EIK) 21(12) (1985) 579–595.
[9] A. Salomaa, Jewels of Formal Language Theory, Pitman, London, 1981.
[10] G.C. Tzeitin, Associative calculus with an unsolvable equivalence problem, Tr. Mat. Inst. Akad. Nauk, 52 (1958) 172–189 (in Russian).