Decidability and Undecidability of Marked PCP

Vesa Halava[1], Mika Hirvensalo[2,1*], and Ronald de Wolf[3,4]

[1] Turku Centre for Computer Science, Lemminkäisenkatu 14 A, 4th floor, FIN-20520, Turku, Finland, vehalava@cs.utu.fi
[2] Department of Mathematics, University of Turku, FIN-20014, Turku, Finland. mikhirve@utu.fi
[3] CWI, P.O. Box 94079, Amsterdam, The Netherlands, rdewolf@cwi.nl
[4] University of Amsterdam

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, 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.

1 Introduction: PCP and Marked PCP

The Post Correspondence Problem (PCP) [6] is one of the most useful undecidable problems, because it can be simply described and many other problems can easily be reduced to it, particularly problems in formal language theory. The general form of the problem is as follows. An instance of PCP is a four-tuple $I = (\Sigma, \Delta, g, h)$, consisting of a finite *source alphabet* $\Sigma = \{a_1, \ldots, a_n\}$, a finite *target alphabet* Δ and two homomorphisms $g, h : \Sigma^* \to \Delta^*$ ($g(ab) = g(a)g(b)$ and $h(ab) = h(a)h(b)$ whenever $a, b \in \Sigma^*$). It is enough to define $g, h : \Sigma \to \Delta^*$, the extension is just concatenation. PCP is the following decision problem:

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

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 general form of this problem is undecidable [6], the reason being that the two morphisms together can simulate the computation of a Turing machine on a specific input. Examining restricted versions of PCP allows one to determine the exact boundary between decidability and undecidability. For instance, the problem becomes trivially decidable (but NP-complete) if we ask for the existence of a solution x of length at most some fixed k [2, p. 228]. If we restrict to g, h which have to be *injective* (g is injective if $x \neq y \Rightarrow g(x) \neq g(y)$), the problem remains undecidable [4]. Also PCP(7), where we restrict to $n = 7$, is

* Supported by the Academy of Finland under grant 14047.

still undecidable [5], but PCP(2) is decidable [1]. As far as we know, decidability or undecidability is still open for $2 < n < 7$.

A further restriction which we will examine in this paper is to have g and h *marked*, which we formally define as follows. If z is a string, we use $Pref_k(z)$ to denote the prefix of length k of z ($Pref_k(z) = z$ if $|z| \leq k$). A homomorphism g is *k-marked* if $g(a)$ and $g(b)$ are nonempty and have $Pref_k(g(a)) \neq Pref_k(g(b))$ whenever $a \neq b \in \Sigma$. An instance $I = (\Sigma, \Delta, g, h)$ of 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 $|\Sigma| \leq |\Delta|$. Without loss of generality we may assume $\Sigma \subseteq \Delta$. Markedness clearly implies injectivity: 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 *generalized* PCP(2). [1] then prove by means of extensive case analysis that marked generalized PCP(2) is decidable. In particular marked PCP(2) is decidable. Here we prove that marked PCP is decidable for *any* alphabet size. We will in fact show that marked PCP is in **EXPTIME** (the class of languages that can be recognized in time upper bounded by $2^{p(N)}$ for some polynomial p of the input size N).

As stated above, PCP can be used for establishing the boundaries between decidability and undecidability. The main result of this paper is decidability of marked PCP. How much can we weaken the markedness condition before we lose decidability? We will show in Section 3 that 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 or 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 biprefix PCP is undecidable [8].[1]

2 Marked PCP Is Decidable

2.1 A Simpler Decision Problem

We would like to give a decision method for marked PCP. First we give an algorithm for the following simpler problem, which also occurs in [1, Section 6]:

> Given marked $I = (\Sigma, \Delta, g, h)$ and $a \in \Delta$, are there $x, y \in \Sigma^+$ such that $g(x) = h(y)$ and $g(x)$ starts with a?

[1] Clearly, a marked morphism is prefix. Both marked and biprefix PCP are special cases of injective PCP, but 2-marked PCP is not. See also at the end of Section 3.

We do not look for $g(x) = h(x)$ here but only for $g(x) = h(y)$, and we additionally require that $g(x)$ starts with some specific $a \in \Delta$. For example, if I has

$$\begin{array}{llll} g(a_1) = a_1 & g(a_2) = a_2 & g(a_3) = a_3a_4 & g(a_4) = a_4 \\ h(a_1) = a_1a_3 & h(a_2) = a_4a_2 & h(a_3) = a_3a_3 & h(a_4) = a_2a_2 \end{array}$$

then for $a = a_1$, a solution would be $x = a_1a_3a_2$ and $y = a_1a_2$.

The next algorithm decides the problem.

1. Set $G = H = \emptyset$, $i = j = 1$.
2. If there are $x_1, y_1 \in \Sigma$ such that $g(x_1)$ and $h(y_1)$ start with a, then set $x = x_1$, $y = y_1$
   else goto 4.
3. (a) If $g(x) = h(y)$, then print "solution $x = x_1 \ldots x_i$ and $y = y_1 \ldots y_j$" and terminate.
   (b) If $g(x)$ is not a prefix of $h(y)$ nor vice versa, then goto 4.
   (c) If $g(x)s = h(y)$, then do the following.
   If $s \in G$ then goto 4; else set $i = i + 1$ and $G = G \cup \{s\}$.
   If there is an x_i such that $g(x_i)$ and s start with the same letter, then set $x = xx_i$ and goto 3; else goto 4.
   (d) If $g(x) = h(y)s$, then analogous to previous step.
4. Print "no solution" and terminate.

Informally, we are building $x = x_1 \ldots x_i$ and $y = y_1 \ldots y_j$, trying to achieve $g(x) = h(y)$. We add on a new x_{i+1} as long as $g(x)$ is a proper prefix of $h(y)$ (i.e., $g(x)s = h(y)$ for some suffix s), and add on a new y_{j+1} if $h(y)$ is a proper prefix of $g(x)$. Note that at each point such x_{i+1} or y_{j+1} are unique (if they exist) because of markedness; if they do not exist we know there is no solution. We keep track of the suffixes we have seen so far in the sets G and H. Because the number of possible suffixes is finite, either the process terminates with a solution, or at some point a suffix is encountered for the second time, in which case we know the process will cycle forever and there is no solution.

The solutions produced by this algorithm are of minimal length. Note carefully that the whole procedure is deterministic, because g and h are marked. Furthermore, if N is the length of the instance I given as input (i.e., the number of bits needed to describe the instance), then this procedure runs in time polynomial in N. Namely, each $g(a_i)$ and $h(a_i)$ can have length at most N, and hence can have at most $N - 1$ proper suffixes. Since there are only $2n = O(N)$ different $g(a_i)$ and $h(a_i)$, there are only $O(N^2)$ different suffixes, hence the loop of the algorithm can be repeated at most $O(N^2)$ times. This loop itself takes $O(N)$ steps, because (1) to check if $g(x) = h(y)$ or $g(x)s = h(y)$ or $g(x) = h(y)s$, we only need to check the way $g(x)$ and $h(y)$ have been changed by the addition of the previous x_i or y_j, and (2) searching for a new x_i (in step c) or y_j (in step d) can be done in $O(n) = O(N)$ steps. Therefore the whole procedure runs in $O(N^3)$ steps.

2.2 Reducing to Simpler Instances

Consider an instance $I = (\Sigma, \Delta, g, h)$ of marked PCP: we have two marked homomorphisms $g, h : \Sigma^+ \to \Delta^+$, where $\Sigma = \{a_1, \ldots, a_n\} \subseteq \Delta$, and we want to decide if there is an $x \in \Sigma^+$ such that $g(x) = h(x)$. Below we describe an approach to decide I by reducing it to an equivalent but simpler instance I' of marked PCP ("equivalent" meaning that I has a solution iff I' has one).

Suppose $\Delta = \{a_1, \ldots, a_l\}$, $l \geq n$. We can run the procedure of the previous section for every $a_i \in \Delta$, yielding pairs of (minimal-length) solutions $(u_1, v_1), \ldots, (u_l, v_l)$ where $u_i, v_i \in \Sigma^+$ and $g(u_i) = h(v_i)$ starts with a_i, or non-existence of solutions for certain i. At most n of the a_i can have a solution. Without loss of generality assume $1, \ldots, m \leq n$ are the i that have a solution. We can turn this into a new instance $I' = (\Sigma', \Delta, g', h')$ of PCP, where $\Sigma' = \{a_1, \ldots, a_m\}$, $g'(a_i) = u_i$ and $h'(a_i) = v_i$. Note that g' and h' are marked, so I' is an instance of marked PCP. Also, since the procedure of the previous section runs in $O(N^3)$ steps and has to be run n times here, I' can be built from I in $O(N^4)$ steps. The reduction from I to I' preserves equivalence:

Lemma 1. *If I and I' are as above, then I and I' are equivalent.*

Proof. Note that every solution x to I must be built up from u_i and v_i: there must be $i_1, \ldots, i_k$ such that $x = u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$. This is easy to see from the example in Figure 1. Here $u_1 = a_5a_3a_1$ and $v_1 = a_5a_3$ is a solution to the simpler problem for a_1, similarly (a_2a_4, a_1a_2) is a solution for a_6 and $(a_6a_3, a_4a_6a_3)$ is a solution for a_2. Here $x = a_5a_3a_1a_2a_4a_6a_3$ is a solution to I, $x' = a_1a_6a_2$ is a solution to I', related by $x = g'(x')$.

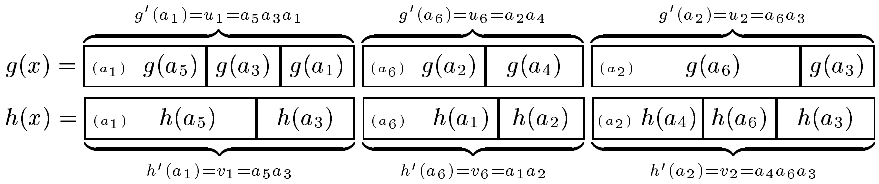

Fig. 1. How a solution to I translates to I' and vice versa

In general, by construction, if x' is a solution to I' then $x = g'(x') = h'(x')$ is a solution to I. And conversely, for every solution x to I there is a solution x' to I' such that $x = g'(x') = h'(x')$. Thus I and I' are equivalent. □

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 eventually decide I. There are at least two ways in which I' can be simpler than I: $|\Sigma'| < |\Sigma|$ ($m < n$) or $\sigma(I') < \sigma(I)$, where σ measures the "suffix complexity" of an instance $I = (\Sigma, \Delta, g, h)$ [1]:

$$\begin{aligned} \sigma(I) = &\,|\cup_{a \in \Sigma} \{x \mid x \text{ is a proper suffix of } g(a)\}| \\ &+ |\cup_{a \in \Sigma} \{x \mid x \text{ is a proper suffix of } h(a)\}| \end{aligned}$$

If $n = m$, we would like I' to be simpler than I in the sense that $\sigma(I') < \sigma(I)$. The following lemma shows that I' at least cannot be *more* complex than I:

Lemma 2. *If I and I' are as above, then $\sigma(I') \leq \sigma(I)$.*

Proof. Define the following four sets:

$$\begin{aligned}
G &= \cup_{a \in \Sigma}\{x \mid x \text{ is a proper suffix of } g(a)\} \\
G' &= \cup_{a \in \Sigma'}\{x \mid x \text{ is a proper suffix of } g'(a)\} \\
H &= \cup_{a \in \Sigma}\{x \mid x \text{ is a proper suffix of } h(a)\} \\
H' &= \cup_{a \in \Sigma'}\{x \mid x \text{ is a proper suffix of } h'(a)\}
\end{aligned}$$

We will define an injective function $p : G' \to H$. Let $u \in G'$, so u is a proper suffix of some specific $g'(a_i) = u_i = x_1 \ldots x_c$ generated by the procedure of the previous section. Let x_r be the first letter of u, and s be the shortest suffix of some $h(y_t)$ due to which x_r was added to u_i in the procedure of the previous section, so s is a prefix of $g(x_r)$ (see Figure 2) or vice versa. Define p as $p(u) = s$.

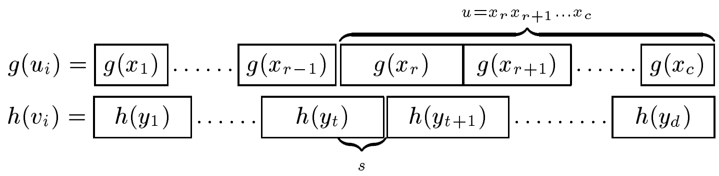

Fig. 2. The suffix s corresponding to u

We will show p is injective. If $u, u' \in G'$ and $p(u) = p(u')$, then u and u' are associated with the same suffix $s = p(u)$, hence u and u' must start with the same x_r and (by determinism of the procedure of the previous section) continue in the same way, giving $u = u'$. Thus p is injective, which implies $|G'| \leq |H|$.

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

2.3 The Algorithm

We will here give a method to decide if a given instance $I = (\Sigma, \Delta, g, h)$ of marked PCP has a solution. The idea is to make a sequence of equivalence-preserving reductions $I_0 = I, I_1, I_2, \ldots$, such that once in a while a reduction from I_i to I_{i+1} simplifies the instance (makes the source alphabet or the suffix complexity smaller). We will show that either this sequence of reductions reaches an I_j which has source alphabet of size 1 or σ equal to 0 (so I_j is decidable), or the sequence will repeat itself after a while and start cycling. Such cycles are detectable, and we will show that every I leading to such a cycle is easily decidable.

So suppose the sequence of reductions does not reach an I_j with alphabet of size 1 or $\sigma(I_j) = 0$. Then it must get "stuck" at a certain source alphabet size and σ. That is, there exist a k, m and z such that all I_i in the infinite sequence $I_k, I_{k+1}, I_{k+2}, \ldots$ have source alphabet of size m and have $\sigma(I_i) = z$. Now this sequence must repeat itself after a while, for otherwise there would be infinitely many distinct instances with the same alphabet and σ-value, contradicting the next lemma.

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

Proof. An instance $I = (\Sigma, \Delta, g, h)$ is completely specified by giving the $2m$ words $g(a_1), \ldots, g(a_m)$, $h(a_1), \ldots, h(a_m) \in \Delta^+$. 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 $z+1$. There are $\sum_{i=1}^{z+1} |\Delta|^i \leq |\Delta|^{z+2}$ such words. Thus there are at most $|\Delta|^{(z+2)2m}$ choices for $2m$ such words, and hence finitely many different I that satisfy $\sigma(I) \leq z$. □

This lemma shows that if the procedure does not converge to very simple instances then it will cycle, and we can detect this by noting that some I_k and I_r $(k < r)$ are equal. It remains to show how we can decide such "cycling" instances of marked PCP. So suppose we have a cycle, assume without loss of generality that it already starts at I_0:

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

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. There must exist some solution x_r to I_r such that

$$x_0 = g_1 g_2 \ldots g_r(x_r)$$
$$x_0 = h_1 h_2 \ldots h_r(x_r)$$

Since the g_i and h_i cannot be length-decreasing, we have $|x_0| \geq |x_r|$. 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 iff I_0 has a 1-letter solution (i.e., there is an $a \in \Sigma_0$ such that $g_0(a) = h_0(a)$), and this is trivially decidable.

Below we summarize this analysis in an algorithm and a theorem:

Decision procedure for marked PCP

1. Set $\mathcal{I} = \emptyset$, $i = 0$, $I_0 = I$.
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 $\mathcal{I} = \emptyset$ and goto 2.
6. If $I_i \in \mathcal{I}$ 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 $\mathcal{I} = \mathcal{I} \cup \{I_i\}$ and goto 2.

Theorem 1. *Marked PCP is decidable.*

2.4 Complexity Analysis

Let us analyze the complexity of this algorithm. Let N be the length of the input instance I. Each reduction from I_i to I_{i+1} can be done in $O(N^4)$ steps. How many different reductions do we need to make? For a fixed alphabet size $|\Sigma| \leq |\Delta| = m$ and suffix complexity z, we can make at most $m^{(z+2)2m}$ reductions before detecting a cycle (proof of Lemma 3). Since $m = O(N)$ and $z = O(N^2)$, this gives an upper bound of $2^{O(\log N \cdot N^3)}$ on the number of reductions for fixed alphabet size and suffix complexity. Alphabet size and suffix complexity cannot increase during the process. There are at most $n = O(N)$ different alphabet sizes and at most $\sigma(I) = O(N^2)$ different suffix complexities possible, so we have to make no more than $O(N^3) \cdot 2^{O(\log N \cdot N^3)}$ reductions. Since the set $\mathcal{I}$ can contain at most $2^{O(\log N \cdot N^3)}$ instances, the test $I_i \in \mathcal{I}$ in step 6 can be performed in $2^{O(\log N \cdot N^3)}$ steps. Thus the whole algorithm works in $2^{O(\log N \cdot N^3)}$ steps, which means that marked PCP is in **EXPTIME**.

3 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 PCP becomes undecidable again.

Consider the following semi-group S_7 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 [7, p. 445]) proved that the following problem for this semi-group is undecidable:

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

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 2-marked PCP. We use a slight modification of the standard reduction from word problems to PCP, 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. Given $u, v \in \Gamma^+$, g and h are defined by Table 1:

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

Table 1. Definition of g and h

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

Lemma 4. *Let u, v, I be as above. Then $u = v \in S_7$ iff I has a solution.*

Proof.
$\Longrightarrow$: Suppose $u = v \in S_7$. 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 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.

$\Longleftarrow$: Suppose I has a solution x. We can assume x is of minimal length. This x must be of the form $Bx_1x_2 \ldots x_mE$, where $x_i \in \Sigma$, so $g(Bx_1 \ldots x_mE) = Bu\#g(x_1 \ldots x_m)E = h(Bx_1 \ldots x_mE) = Bh(x_1 \ldots x_m)\underline{\#}vE$. 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 $1 \leq i \leq k-1$, from which $u = v \in S_7$ follows.

Because $\#$ occurs in $h(x_1 \ldots x_m)$, there must be some least 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

$x_1, \dots, x_{i-1}$ must have been chosen from $a, \dots, e, r_1, \dots, r_7$. Let $x_1 \dots x_{i-1} = w_1 r_{i_1} w_2 r_{i_2} \dots w_l$, with $w_i \in \Gamma^*$ and $r_i = [s_i = t_i] \in \{r_1, \dots, r_7\}$. Then $u = h(w_1 r_{i_1} w_2 r_{i_2} \dots w_l) = w_1 s_{i_1} w_2 s_{i_2} \dots w_l$. See Figure 3 for illustration.

$$g(Bx_1 \dots x_i \dots x_m E) = \overbrace{\boxed{B \quad w_1 s_{i_1} w_2 s_{i_2} \dots w_l \quad \#}}^{g(B)=Bu\#}\boxed{g(x_1)}\boxed{g(x_2)}\dots\dots\overbrace{\boxed{E}}^{g(E)}$$

$$h(Bx_1 \dots x_i \dots x_m E) = \underbrace{\boxed{B}}_{h(B)}\ \underbrace{\boxed{w_1 s_{i_1} w_2 s_{i_2} \dots w_l}}_{h(x_1 \dots x_{i-1})=u=u_1}\ \underbrace{\boxed{\#}}_{h(x_i)}\ \boxed{h(x_{i+1})}\dots\dots\underbrace{\boxed{\underline{\#}vE}}_{h(E)}$$

Fig. 3. Picture leading to $u = v$

Note that $g(x_1 \dots x_{i-1}) = g(w_1 r_{i_1} w_2 r_{i_2} \dots w_l) = \underline{w_1 t_{i_1} w_2 t_{i_2} \dots w_l}$. But now, since we must have $g(x_1 \dots x_m E) = h(x_{i+1} \dots x_m E)$, there must be a least $j > i$ such that $x_j \in \{\#, \underline{\#}\}$ and $h(x_{i+1} \dots x_{j-1}) = g(x_1 \dots x_{i-1}) = \underline{w_1 t_{i_1} w_2 t_{i_2} \dots w_l}$. 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 $u = v \in S_7$, since $g(x)$ starts with $u_1 = u$ and ends with $u_k = v$. □

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

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

To end this section, we note that 2-marked PCP is not a special case of 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 strongly 5-marked PCP is undecidable: the biprefix instances of PCP constructed there to show undecidability of biprefix PCP are also 5-marked. Decidability of strongly k-marked PCP for $1 < k < 5$ is still open.

4 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 PCP to *marked* morphisms gives us decidability. On the other hand, 2-marked PCP is still undecidable.

The following questions are left open by this research:

- Is exponential time the best we can do when deciding marked PCP, or is there a polynomial-time algorithm for the problem?
- What about decidability of strongly k-marked PCP for $1 < k < 5$?
- What about decidability of marked *generalized* PCP [1,3]?
- The decidability status of 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. Marked PCP is a subcase of elementary PCP which we have shown here to be decidable. Can our results help to settle the decidability status of elementary PCP?

Acknowledgment

We would like to thank Tero Harju, Juhani Karhumäki, and John Tromp for reading and commenting this paper, and Harry Buhrman for some discussions. 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, and G. Rozenberg. The (generalized) Post correspondence problem with lists consisting of two words is decidable. *Theoretical Computer Science*, 21(2):119–144, 1982.
2. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
3. T. Harju, J. Karhumäki, and D. Krob. Remarks on generalized Post correspondence problem. In *Proceedings of 13th STACS*, volume 1046 of *Lecture Notes in Computer Science*, pages 39–48. Springer-Verlag, 1996.
4. 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*, 257:2940–2943, 1963.
5. Y. Matiyasevich and G. Sénizergues. Decision problems for semi-Thue systems with a few rules. In *Proceedings of the 11th IEEE Symposium on Logic in Computer Science*, pages 523–531, 1996.
6. E. L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52:264–268, 1946.
7. G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1. Springer-Verlag, Berlin, 1997.
8. K. Ruohonen. Reversible machines and Post's correspondence problem for biprefix morphisms. *Journal of Information Processing and Cybernetics (EIK)*, 21(12):579–595, 1985.
9. A. Salomaa. *Jewels of Formal Language Theory.* Pitman, 1981.
10. G. C. Tzeitin. Associative calculus with an unsolvable equivalence problem. *Tr. Mat. Inst. Akad. Nauk*, 52:172–189, 1958. In Russian.