Simple Words in Equality Sets

Marjo Lipponen
Arto Salomaa

Turku Centre for Computer Science
TUCS Technical Report No 39
August 1996
ISBN 951-650-817-0
ISSN 1239-1891

Abstract

It is well known that equality sets between two morphisms possess a remarkable generative capacity: an arbitrary recursively enumerable set is obtained from an equality set by certain simple operations. Interconnections between simplicity of computations and structural primitivity of words in equality sets have also been observed. The paper discusses recent work in this area, pointing out certain open problems and emphasizing new directions for research.

TUCS Research Group
Mathematical Structures of Computer Science

1 Post Correspondence Problem and expressibility of equality sets

We view an instance of the *Post Correspondence Problem*, *PCP*, as a pair (g,h) of morphisms $g,h:\Sigma^*\to\Delta^*$. A *solution* for this instance (g,h) is any word in the *equality set* (or *equality language*)

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

Thus, the equality set being empty means that there are no solutions; the empty word λ is not considered to be a solution.

For example, consider the range and target alphabets $\Sigma=\{1,2,3\}$ and $\Delta=\{a,b\}$ and define an instance (g,h) of the PCP by the table

	1	2	3
g	a^2	b^2	ab^2
h	a^2b	ba	b

Every solution must begin with the letter 1, after which 2, 1 and 3 must follow, in this order. This is due to the fact that, for any word w', whenever w' begins a solution w, $w=w'x$, then one of the words $g(w')$ and $h(w')$ must also be a prefix of the other. For our instance (g,h), the situation can be described by the table

w'	1	12	121	1213
$g(w')$	a^2	a^2b^2	$a^2b^2a^2$	$a^2b^2a^3b^2$
$h(w')$	a^2b	a^2b^2a	$a^2b^2a^3b$	$a^2b^2a^3b^2$

The table shows that $w'=1213$ is actually a solution. Could we build up the word w' in a different way? Having in mind the above-mentioned prefix condition concerning $g(w')$ and $h(w')$, we see that w' must begin with 12. But after that we have also the possibility 3 because $h(123)=a^2b^2ab$ is a prefix of $g(123)=a^2b^2ab^2$. After this choice, our only further possibility is 3, leading to the words $h(1233)=a^2b^2ab^2$ and $g(1233)=a^2b^2ab^2ab^2$. But now this road is blocked; there is no way we can continue. We should take care of the excess part ab^2 using the h-words – but there is no word x such that ab^2 is a prefix of $h(x)$. This analysis shows that the choices indicated in the above table are the only ones leading to a solution.

Clearly, an arbitrary catenation of solutions is again a solution. Thus, we have characterized all solutions for this particular instance (g,h) of the PCP:

$$E(g,h)=(1213)^+.$$

It is obvious that the set of solutions is always a *star language*: whenever $L = E(g, h)$ for some (g, h), then $L = L^+$. If a solution w is a product of some other solutions, say $w = w_1 w_2 w_3$, it is natural to regard the factors w_i as "simpler" or "more primitive" solutions. Indeed, various types of "prime solutions" can be defined. The purpose of the present article is to survey recent work carried out along these lines, as well as to point out some further directions and problems. Apart from sheding light on equality sets and the PCP in general, such primality considerations have intrinsic connections with certain aspects of combinatorics on words, as will become apparent below.

Let us still go back to the instance (g, h) discussed above. It is seen that, on the road to success, the morphism h "runs faster" than g: for any prefix w' of a solution, $g(w')$ is a prefix of $h(w')$. When the morphism g "catches up" the morphism h, it means that a solution is found. Exactly the same idea is used in establishing general representability results in terms of equality sets. One builds up a word describing a computation or a derivation step by step, using two morphisms. One of them runs faster. If the other catches up, which happens for words in the equality set, it means that such a computation or derivation is actually possible. To get the representability result, the final result of the computation or derivation has still to be "squeezed out" from the long word obtained.

Let us become more specific and prove that an arbitrarily given recursively enumerable language $L \subseteq T^*$ can be represented in the form

$$L = h_T(E(g, h) \cap R), \tag{1}$$

for some morphisms g, h, regular language R and *weak identity* h_T. (That is, h_T is a morphism mapping the letters of T into themselves and erasing all other letters.) The proof use the ideas indicated above and will refer to derivations and grammars. A similar construction referring to computations and Turing machines is given in Lemma 3.1 of [MSSY].

Thus, let $L \subseteq T^*$ be a language generated by a (type 0) grammar $G = (N, T, S, P)$, where the production set P consists of the productions

$$p_i : \alpha_i \to \beta_i, \quad 1 \le i \le n.$$

(Observe that we associate a label p_i to each production.) We consider also "primed versions" of letters by defining $T' = \{a' \mid a \in T\}$. The domain and target alphabets of our two morphisms g and h will be

$$\begin{aligned} \Sigma &= \Delta \cup P \cup T' \cup \{B, F\}, \\ \Delta &= N \cup T \cup \{\Rightarrow\}, \end{aligned}$$

where, intuitively, B and F are beginning and final letters and $\Rightarrow$ marks the yield relation between two derivation steps. The two morphisms themselves are defined by the table

	B	$\Rightarrow$	$p_i \in P\ (p_i : \alpha_i \to \beta_i)$	$A \in N$	$a' \in T'$	$a \in T$	F
g	$S \Rightarrow$	$\Rightarrow$	β_i	A	a	λ	λ
h	λ	$\Rightarrow$	α_i	A	a	a	$\Rightarrow$

Further, we define the regular language R by

$$R = B(V^*PV^* \Rightarrow)^+T^*F, \quad \text{where } V = N \cup T'.$$

We omit the detailed argument showing that (1) is valid – the interested reader is referred to [Sa1]. The construction follows ideas presented above. The operation $\cap R$ restricts the possibilities to words of $E(g,h)$ resembling derivations. The word being in $E(g,h)$ guarantees that the derivation is properly performed. The morphism g runs faster than h, and h_T erases everything except the final result of the derivation. Specifically, whenever

$$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_k$$

is a derivation according to G, then there is a word $w \in B(V^*PV^* \Rightarrow)^+$ such that

$$g(w) = S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_k \Rightarrow,$$
$$h(w) = S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_{k-1} \Rightarrow .$$

Moreover, if $\alpha_k = x \in T^*$, then

$$wxF \in E(g,h) \cap R \quad \text{and} \quad h_T(wxF) = x.$$

The operation $\cap R$ is needed to rule out unwanted words. Repetitions of a successful derivation of x give rise to a word in the equality set $E(g,h)$ and consequently, without the operation $\cap R$, we would get all powers x^i together with x. This can be avoided, as done in [C], if only words from $E(g,h)$ are taken into account such that no proper prefix of the word is in $E(g,h)$. Such solutions of the instance (g,h) are called *F-prime* in the sequel. Thus, the representability result (1) becomes simpler if the set of all solutions $E(g,h)$ is replaced by the set of F-prime solutions. This is one of the early reasons for investigating primitive solutions of the PCP. Further interconnections between the primitivity of solutions and complexity of computations are indicated in [MS2] and [MSSY].

In the representation result (1), the "squeezing" mechanisms $\cap R$ and h_T can be expressed as one *gsm-mapping* f: $L = f(E(g,h))$. This follows because both $\cap R$ and h_T are gsm-mappings, and gsm-mappings are closed under composition. (We note in passing that the abbreviation "gsm" for "generalized sequential machine" was well known already in the early 70's, whereas nowadays the principal meaning of "gsm" is quite different. Can the reader guess what the principal meaning of the abbreviation "PCP" is? Several dictionaries seem to agree in this respect – none of them knows "Post Correspondence Problem"!)

The representation result (1) can be modified to a form that is particularly interesting in view of the recent studies concerning DNA-computing. Consider the binary alphabet $\Sigma_2 = \{0,1\}$ and its barred version $\overline{\Sigma}_2 = \{\overline{0}, \overline{1}\}$. The barred version $\overline{w}$ of a word $w \in \Sigma_2^*$ is obtained by barring every letter of w. The *twin-shuffle language* L_{ts} over the alphabet $\Sigma_2 \cup \overline{\Sigma}_2$ is defined by

$$L_{ts} = \{w \in (\Sigma_2 \cup \overline{\Sigma}_2)^* \mid \overline{h_{\Sigma_2}(w)} = h_{\overline{\Sigma}_2}(w)\}.$$

Thus, L_{ts} consists of words obtained by taking an arbitrary word $x \in \Sigma_2^*$, its barred version $\overline{x} \in \overline{\Sigma}_2^*$ and shuffling the two in a fashion otherwise arbitrary, but preserving the order of letters in both x and $\overline{x}$. (The reader is referred to [MS3] for the basics of the shuffle operation.) The twin-shuffle language can be defined also as an equality set. Let $h'_{\Sigma_2} : (\Sigma_2 \cup \overline{\Sigma}_2)^* \to \overline{\Sigma}_2^*$ be the morphism defined by

$$h'_{\Sigma_2}(a) = \begin{cases} \lambda & \text{for } a \in \Sigma_2, \\ b & \text{for } a = \overline{b} \in \overline{\Sigma}_2. \end{cases}$$

Then $L_{ts} = E(h_{\Sigma_2}, h'_{\Sigma_2}) \cup \{\lambda\}$. Instead of starting with the binary alphabet Σ_2, we can define in a similar way twin-shuffle languages for other alphabets as well.

The representation result (1) assumes now the following form. Let again $L \subseteq T^*$ be an arbitrary recursively enumerable language. Then there is a gsm-mapping f_L such that

$$L = f_L(L_{ts}). \tag{2}$$

Thus, the role of L_{ts} is universal. Everything specific about the given language L is implemented in the gsm-mapping f_L. The representation (2) is obtained from (1) roughly as follows. ([Sa1, pp. 114–117] contains further details.)

The restrictive capacity of the intersection $E(g,h) \cap R$ can be simulated by replacing the letters b of R with the words $g(b)\overline{h(b)}$, and then intersecting the resulting regular language $R(g,h)$ with a twin-shuffle language L'_{ts}. Originally, this language L'_{ts} may involve more than two letters but it can be obtained from the binary twin-shuffle language L_{ts} by an inverse gsm-mapping. Thus, we are applying to L_{ts} the operations of (i) an inverse gsm-mapping, (ii) an intersection with the regular language $R(g,h)$ and (iii) a weak identity, whose composition amounts to one gsm-mapping f_L. A (nonempty) word w is in the equality set $E(g,h)$ exactly in case the word $g(w)\overline{h(w)}$ is in the (appropriate) twin-shuffle language. This is the key idea.

This key idea also clearly points out the interconnection with DNA-computing. DNA consists of four bases, customarily denoted A (adenine), C (cytosine), G (guanine) and T (thymine). The chemical structure of these molecules causes a strong affinity between A and T, and likewise between C and G. This affinity allows a string of bases, say TAGCAT, to pair up with its complementary string ATCGTA, customarily referred to as its *Watson-Crick complement*, the result being a double-stranded molecule, $\frac{\mathrm{TAGCAT}}{\mathrm{ATCGTA}}$. (We have somewhat simplified the matters here by not speaking of the polarity of the strings, which is unessential for us. We refer to [Lila] for further details.) Let us denote A, T, C, G by 0, $\overline{0}$, 1, $\overline{1}$, respectively. Let us, further, rewrite double strands as one-dimensional strings by taking characters from both strands by turns. Thus, our example $\frac{\mathrm{TAGCAT}}{\mathrm{ATCGTA}}$ becomes first $\frac{\overline{0}0\overline{1}10\overline{0}}{0\overline{0}1\overline{1}\overline{0}0}$ and then $\overline{0}00\overline{0}\overline{1}11\overline{1}0\overline{0}\overline{0}0$, a word in the twin-shuffle language! In this way Watson-Crick complementarity yields words in the twin-shuffle language. (All words in L_{ts} are not obtained in this fashion.) So far many models have been proposed for DNA-computing. In all of them Watson-Crick complementarity plays a central role. The relation (2) guarantees the universality of computations in all models, provided the input-output format is sufficient to take care of the role of the gsm-mapping f_L in (2). Further discussion about this matter lies beyond the scope of this article.

We conclude this introductory section with a further example. Consider the instance (g_1, h_1) of the PCP defined by the table

	1	2	3
g_1	b^2	ab	c
h_1	b	ba	bc

This example is a modification of the one in Post's seminal paper [Post]. Clearly, $E(g_1, h_1) = (12^*3)^+$. Moreover, the set of F-prime solutions (in the sense indicated above) consists of all words in 12^*3.

It is well known that PCP is one of the "cornerstones of undecidability", [RS], customarily used when a language-theoretic problem is shown to be undecidable. The simplest way of showing that PCP itself is undecidable is to use Post normal systems, see [Post, Sa2, RS].

2 Prime solutions and primality types

We have illustrated the expressive capabilies of equality sets from many different angles. We wanted in this way to prepare background for primality considerations, the actual theme of this article. We already pointed out that some solutions of (an instance of) the PCP, that is, some words in the equality set, are "simpler" or "more primitive" than others. We will investigate various classes of such prime solutions, surveying recent work done in this area. It is clear that the structural simplicity of a word in an equality set is reflected also in representation results based on equality sets, a typical example being the result by Culik, [C], already mentioned above. However, many details concerning the significance of various primality aspects to the representation of computations are not yet properly understood and remain to be investigated further. In this respect it is interesting to note the fact established in [MSSY] that deterministic computations correspond to morphisms of bounded delay.

Let us still go back to the instance (g_1, h_1) introduced at the end of Section 1. The words in the regular language 12^*3 are solutions. Moreover, no proper final subword can be removed from such a word and still have a solution. However, words of the form 12^i3, $i \geq 1$, are not "prime solutions" in view of the fact that the subword 2^i, or some part thereof, can be removed and still have a solution. This leads to the notion of an *S-prime* solution: no proper subword can be removed and still have a solution. In this case 13 is the only S-prime solution. A still more restrictive notion is that of a *P-prime* solution: no scattered subword can be removed and still have a solution.

Let us do these definitions formally. Consider morphisms $g, h : \Sigma^* \rightarrow \Delta^*$. For a word w over Σ^*, we consider three sets of words obtained from w by removing a final subword, a subword or a scattered subword, respectively. More specifically, define

$$\begin{aligned} fin(w) &= \{v \mid w = vx, \text{ for some } x \in \Sigma^*\}, \\ sub(w) &= \{v_1v_2 \mid w = v_1xv_2, \text{ for some } v_1, v_2, x \in \Sigma^*\}, \\ scatsub(w) &= \{v_1 \dots v_k \mid w = x_1v_1 \dots x_kv_kx_{k+1}, \text{ for some } x_i, v_i \in \Sigma^*\}. \end{aligned}$$

We want to emphasize that $sub(w)$ is obtained by catenating the *remaining parts* when a subword of w is removed. The set $scatsub(w)$ can be viewed in an analogous way but also as the set of scattered subwords of w.

We can now determine three subsets of the equality set,

$$\begin{aligned} F(g,h) &= \{w \in E(g,h) \mid fin(w) \cap E(g,h) = \{w\}\}, \\ S(g,h) &= \{w \in E(g,h) \mid sub(w) \cap E(g,h) = \{w\}\}, \\ P(g,h) &= \{w \in E(g,h) \mid scatsub(w) \cap E(g,h) = \{w\}\}. \end{aligned}$$

Words in these three sets are called *F-prime*, *S-prime* and *P-prime solutions* for the instance (g,h).

It follows immediately that every P-prime is an S-prime, and every S-prime is an F-prime, i.e.,

$$P(g,h) \subseteq S(g,h) \subseteq F(g,h) \subsetneq E(g,h).$$

One or both of the first inclusions may be strict whereas the third inclusion is always strict, provided $E(g,h)$ is nonempty. If $E(g,h)$ is nonempty then so must be the three other sets. In addition, each of the four sets is recursive.

The triple (p,s,f), where p, s and f are the cardinalities of the sets $P(g,h)$, $S(g,h)$ and $F(g,h)$, respectively, is defined to be the *primality type* of the instance (g,h). Thus p, s and f are nonnegative integers or ∞.

For the specific instance (g,h) considered at the beginning of Section 1, we have $E(g,h) = (1213)^+$ and

$$P(g,h) = S(g,h) = F(g,h) = \{1213\}.$$

Thus, the primality type of this instance is $(1,1,1)$. The primality type of the instance (g_1,h_1) considered at the end of Section 1 is $(1,1,\infty)$.

A complete characterization of primality types was given in [SSY]. It is a straightforward consequence of the definition that if one of the components p,s,f equals zero, all three components must equal zero. Similarly, we have always $p \leq s \leq f$. According to a result originally due to Higman (see [Hi, MS3]), every language L, with the property that no word in L is a scattered subword of another word in L, is finite. This implies that the component p is finite in every primality type.

A further restriction on primality types (p,s,f) is due to the following result established in [SSY]. Assume that the word xyz is in $F(g,h)$, where x,y,z are nonempty, and that xz is in $E(g,h)$. Then xy^iz is in $F(g,h)$, for all $i \geq 0$.

It is a consequence of this result that, whenever $s < f$, then $f = \infty$. There are no further restrictions on primality types (p,s,f). Indeed, given any triple (p,s,f) such that either $p = s = f = 0$, or else (i) $1 \leq p \leq s \leq f$, (ii) p is finite and (iii) if $s < f$ then $f = \infty$, an instance (g,h) having (p,s,f) as its primality type can be effectively constructed. Details are given in [SSY].

We now take a step further and consider prime words over the target alphabet Δ in a similar way. We first introduce the "co-sets" corresponding to the sets $E(g,h)$, $F(g,h)$, $S(g,h)$ and $P(g,h)$:

$$co\text{-}E(g,h) = \{\alpha \in \Delta^* \mid g(w) = h(w) = \alpha, \text{ for some } w \in \Sigma^+\},$$

$$co\text{-}F(g,h) = \{\alpha \in co\text{-}E(g,h) \mid fin(\alpha) \cap co\text{-}E(g,h) = \{\alpha\}\},$$
$$co\text{-}S(g,h) = \{\alpha \in co\text{-}E(g,h) \mid sub(\alpha) \cap co\text{-}E(g,h) = \{\alpha\}\},$$
$$co\text{-}P(g,h) = \{\alpha \in co\text{-}E(g,h) \mid scatsub(\alpha) \cap co\text{-}E(g,h) = \{\alpha\}\}.$$

Words in the last three sets are called *F_c-prime*, *S_c-prime* and *P_c-prime* with respect to the instance (g,h). Thus, a word α over Δ being an F_c-prime means that α is the morphic image (under g and h) of a solution and, moreover, no proper prefix of α has this property. Whereas

$$co\text{-}E(g,h) = g(E(g,h)) = h(E(g,h)),$$

we have only

$$co\text{-}F(g,h) \subseteq g(F(g,h)) = h(F(g,h))$$

and, moreover, the inclusion is strict for some instances (g,h). The situation is analogous as regards the sets $co\text{-}S(g,h)$ and $co\text{-}P(g,h)$.

Let p_c, s_c and f_c be the cardinalities of the sets $co\text{-}P(g,h)$, $co\text{-}S(g,h)$ and $co\text{-}F(g,h)$, respectively. The six-tuple (p_c, p, s_c, s, f_c, f) is defined to be the *mixed primality type* of the instance (g,h).

Again the inequalities

$$p_c \leq s_c \leq f_c, \quad p_c \leq p, \quad s_c \leq s, \quad f_c \leq f$$

are easily established. A further restriction on mixed primality types is the following unexpected fact, established in [MS2]. If $s = 1$, then $f_c = 1$ or $f_c = \infty$.

These are the only restrictions on mixed primality types. To summarize, let (p_c, p, s_c, s, f_c, f) be a six-tuple of *positive* integers or ∞ satisfying each of the following conditions (i)-(vi):

(i) $p \leq s \leq f$,

(ii) $p_c \leq s_c \leq f_c$,

(iii) $p_c \leq p,\ s_c \leq s,\ f_c \leq f$,

(iv) p is finite,

(v) if $s < f$, then $f = \infty$,

(vi) if $s = 1$, then $f_c = 1$ or $f_c = \infty$.

Then an instance (g,h) with this six-tuple as its mixed primality type can be effectively constructed. All details of the construction are given in [MS2].

We have, thus, an exhaustive characterization of both primality types and mixed primality types.

We will view next the problem of primality from the opposite angle, that is, the procedure is reversed. Given a word w, we search for an instance (g,h) of the PCP such that w is an F-prime (resp. S-prime, P-prime) solution of (g,h).

3 P-, S- and F-words

The study of prime words, initiated in [MS1], has turned out to be especially interesting from point of view of combinatorics on words. In this section we take a closer look at the properties of P-, S- and F-words.

By applying Makanin's result, [Ma], of the solvability of a finite system of equations, we can prove that each of the properties of being a P-word, an S-word or an F-word is decidable. Nevertheless, because of the difficulty of Makanin's general algorithm we gain very little information of the nature of these prime words. Therefore we have tried to seek other methods to characterize them.

The following two notions are important in the sequel. We say that u is *Parikh shorter* than v if $\psi(u) \leq \psi(v)$ componentwise. The *basic Parikh vector* ψ_0 of w is obtained from the Parikh vector $\psi(w) = (x_1, \ldots, x_n)$ by dividing it with $\gcd(x_1, \ldots, x_n)$. Thus the components of the basic Parikh vector are always relatively prime. (Notice that this is not the same as being pairwise relatively prime, as seen from the vector $(2,6,9)$.)

The main result in [Li3] says that for each word w we can effectively find an instance (g,h) such that w is a solution and any other word w' which is Parikh shorter than w is also a solution if and only if w and w' have the same basic Parikh vectors, $\psi_0(w) = \psi_0(w')$.

Let us consider an example. For a word $w = 112321323$ we have $\psi(w) = (3,3,3)$ and $\psi_0(w) = (1,1,1)$. Thus by the previous result we can find an instance (g,h) such that w and all the words whose Parikh vectors are $(2,2,2)$ or $(1,1,1)$ (such as 132312 and 312) are solutions but, for instance, $112, 1123$ or any other proper prefix of w is not a solution for this instance (g,h). But this means that w is an F-prime solution and hence an F-word. An instance showing this is the following.

	1	2	3
g	a^{21}	a	a
h	a	a^5	a^{17}

Let us call a word *ratioprimitive* if none of its prefixes has the same basic Parikh vector as the whole word. We can easily show that F-words are exactly ratioprimitive words. Hence we have obtained an effective algorithm for F-words: it is easy to check for a given word whether it is ratioprimitive or not. For instance, the words 112321323 and *night* are F-words while 112323123 and *noon* are not F-words.

For S- and P-words the results obtained so far are not as strong. Using the result above, we can prove that any subratioprimitive word is also an S-word where a word is defined *subratioprimitive* when none of its subwords has the same basic Parikh vector as the whole word. Moreover, any word whose Parikh vector is the same as its basic Parikh vector, i.e., its components are relatively prime, is a P-word. Another subclass of P-words, found in [MS1], are the words of the form $a_1^{i_1} a_2^{i_2} \dots a_n^{i_n}$ where $\Sigma = \{a_1, \dots, a_n\}$ and $n \geq 2$. Thus, for instance, 12321 and 113322 are P-words.

We can prove that these properties do not characterize P-words nor S-words completely. For instance, 311132223 is an S-word but not subratioprimitive ($\psi_0(132) = \psi_0(311132223)$) and 122313 is a P-word though $\psi(122313) = 2(1,1,1)$. These claims are established by the following instances

	1	2	3
g_1	a^6b	a	a^2
h_1	a^2	ba^6	a

	1	2	3
g_2	a	a^8b	a
h_2	a^2	a^2	a^3ba^3

Nevertheless, we conjecture that in a binary alphabet $\{a, b\}$ a nonempty word w is a P-word and an S-word if and only if $\psi(w) = \psi_0(w)$ or $w \in a^+b^+ \cup b^+a^+$.

One way to prove this conjecture is to show that words that are not *P-type*, that is $\psi(w) \neq \psi_0(w)$ and $w \notin a^+b^+ \cup b^+a^+$, must be *periodicity forcing*. In other words, every instance (g, h) for which they are solutions is *trivial* (for some $a \in \Sigma$, $g(a) = h(a)$) or *periodic* (for some $u \in \Delta^+$, $g(a) = u^{i_a}$, $h(a) = u^{j_a}$ for all $a \in \Sigma$). Intuitively, as the name says, these words force their corresponding instances to be periodic (or trivial). We have a lot of examples supporting this conjecture but no formal proof. For instance, the words w where $\psi(w) = k(1,1,1)$, $2 \leq k \leq 4$, and $w \notin a^+b^+ \cup b^+a^+$ are all periodicity forcing and hence not P- and S-words.

Also in larger alphabets (with at least three letters) we can use the property of being periodicity forcing to characterize words that are not P- or S-words. It is easy to show that if a periodicity forcing word is not subratioprimivite (resp. P-type) then it cannot be an S-word (resp. a P-word). Thus the study of periodicity forcing words can shed new light on the characterization of P- and S-words. But not completely since there are non-S-words, such as 1212123123, which are not subratioprimitive but not periodicity forcing either.

Our attempt to characterize prime words benifits also from the fact that the properties of not being a P-word or an S-word are preserved by nonerasing morphisms. In this way we can enlarge the sets of non-P-words and non-S-words found so far. For instance, since 112122 is not a P-word also 112312323 and 121231233 are not P-words.

Next we study the hierarchy between P-, S- and F-words. It follows from the definition that every P-word is also an S-word and every S-word is also an F-word. So we only need to settle whether these inclusions are proper or not.

The relation between S- and F-words turns out to be strict in every alphabet. We can prove that each word $w = (123\ldots(k-1))^2(123\ldots k)^2$ over a k-letter alphabet, $k \geq 2$, is an F-word since it is ratioprimitive but not an S-word because every time w is a solution for some instance (g,h) then also $w' = 123\ldots(k-1)123\ldots k$ is a solution and $w' \in sub(w)$.

For P- and S-words the result is slightly weaker. We can prove that if the alphabet has at least three letters then the relation is proper. In a three-letter alphabet the word $w = a^2b^3c^3a^4b^3c^3a^3$ is an S-word, being subratioprimitive, but not a P-word since it is periodicity forcing and not P-type (we have $\psi(w) = 3(3,2,2)$). In bigger alphabets we, respectively, consider the morphic images of the word w under the morphism

$$\phi(a) = a, \quad \phi(b) = b, \quad \phi(c) = c_1\ldots c_n.$$

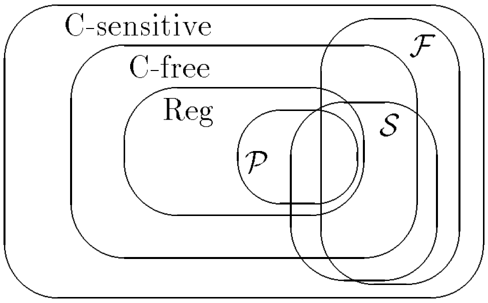
As mentioned above $\phi(w)$ cannot be a P-word since w is not a P-word but it is still subratioprimitive and, hence, an S-word.

The only case that still remains open is the binary alphabet for P- and S-words. We conjecture that in this case P- and S-words are actually equal and, what is more, the sets of P-prime and S-prime solutions are the same for all instances (g,h).

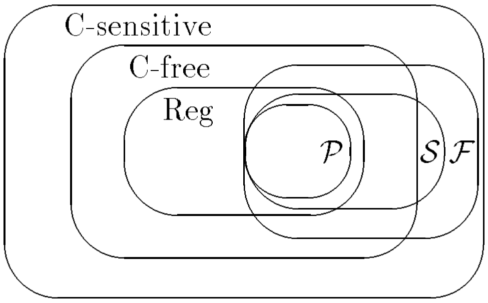
In fact, we can prove that if at least one of the morphisms g and h is periodic then $P(g,h) = S(g,h)$. This follows from the characterization of the equality sets in these types of instances. On the other hand, if g and h are both injective then the equality set is either of the form $\{u,v\}^+$, $\{w\}^+$ or $(\beta\gamma^*\delta)^+$ (see [EKR] for details) and, hence, in the last two cases the sets of S-prime and P-prime solutions are again equal. The case $\{u,v\}^+$ seems also be clear, at least the only known examples of equality sets generated by two words are $\{1^i2, 21^i\}^+$ for some fixed $i \geq 1$ and here $P(g,h) = \{1^i2, 21^i\} = S(g,h)$. An interesting observation is that the conjecture concerning periodicity forcing words (every non-P-type is periodicity forcing) would prove this conjecture, too.

The end of this section is devoted to prime languages of the Post Correspondence Problem. There are two types of definitions, based on equality or inclusion. We say that L is an *F-language* in *equation sense* if, for some instance (g,h), $L = F(g,h)$ and in *inclusion sense* if $L \subseteq F(g,h)$.

The following two figures show how prime languages lie in the Chomsky hierarchy. In the first figure we use the equation method. Notice that in this case P-, S- and F-languages do not form an increasing hierarchy.



The second picture settles the relation for the inclusion method.



Examples of all areas in these pictures can be given.

[Li4] studies also other properties of prime languages. Using Makanin's result, [Ma], as for prime words, we can prove that for a finite language L it is decidable whether L is a P-language, an S-language or an F-language in inclusion sense. In this type of prime languages we can actually characterize F-languages completely in a binary alphabet since L is an F-language if and only if every element of L is ratioprimitive and everyone has the same basic Parikh vector. For S- and P-languages we can only say that if every element of L has the same Parikh vector whose components are relatively prime or $L \subseteq \{a^i b, ba^i\}$ for some $i \geq 1$, then L is a P-language and an S-language.

The main characterization result for P- and S-languages in equation sense is that if L has at least three elements then L is a P-language and an S-language in a binary alphabet if and only if L is of the form $c(\{a^i b^j\})(i,j) = 1)$.

The set $c(\{a^ib^j\})(i,j) = 1)$ consists of all the words whose Parikh vector is (i,j) and i and j are relatively prime. For instance, $c(\{a^3b^1\}) = \{aaab, aaba, abaa, baaa\}$. For F-languages in equation sense we can only say that if L is finite and it has at least three elements then L cannot be an F-language.

Notice that also in connection of prime languages periodicity forcing words play an important role. For instance, the hierarchy of P-, S- and F-words in equation sense was proved not to be increasing, using periodicity forcing words.

A further interesting topic of prime languages is the finiteness of P-languages. This follows from the result of the second section where $P(g,h)$ was proved to be finite for every instance (g,h). Thus the question arises whether the maximal number of P-prime solutions can be counted in terms of the cardinality of the domain alphabet Σ and the size of the instance (g,h), that is $\max\{|g(a_i)|, |h(a_i)| : a_i \in \Sigma\}$. In a binary alphabet this has been completely settled in [Li4], the maximal number being

$$\frac{[(l-2)+(l-1)]!}{(l-2)!(l-1)!}$$

where the size of the instance is at most l. Thus if $l = 5$, the maximal number is 35.

In larger alphabets the question is settled for periodic instances and we conjecture that this is also the maximum in every instance. The algorithm is, however, too long to be presented here.

4 Variations

So far we have studied only three types of prime solutions based on removing a suffix, a subword or a scattered subword. In [Li4] also other ways to define the primality were studied.

We start with strongly prime PCP-words, based on [LP]. This means that words are primitive for *every* (not only some) instance (g,h) for which they are solutions. This definition turns out to be too restrictive (even after leaving out instances (g,h) where $g = h$) since in alphabets with at least three letters no word is strongly prime: for every word we are able to find an instance for which the word is not primitive. In a binary alphabet, however, we have managed to completely characterize these new types of words. We can prove that any word w is strongly P-prime and S-prime if and only if its Parikh vector is the same as its basic Parikh vector, $\psi(w) = \psi_0(w)$, whereas any strongly F-prime is also F-prime and vice versa. Thus if strongly prime words are denoted with upper primes we obtain the following hierarchy

$$P_2' = S_2' \subset P_2 \subseteq S_2 \subset F_2 = F_2'$$

where all the inclusions are proper except possibly $P_2 = S_2$ as already mentioned in the previous section.

For many purposes it is useful to regard primitive solutions as obtained from the equality set $E(g,h)$ by certain operations, such as partial orders. We now study solutions that are minimal with respect to the given partial order. For instance, $P(g,h)$ is obtained from $E(g,h)$ by $\leq_s$ where $u \leq_s w$ if u is a scattered subword of w and $F(g,h)$ is obtained by $\leq_p$ where $u \leq_p w$ if u is a prefix of w. In the same way we obtain new primitive solutions by comparing their lengths, Parikh vectors, lexicographic order and so on. The primitive solutions obtained by the first-mentioned two partial orders (length, Parikh vector) are especially interesting since their cardinalities μ and ψ in every instance (g,h) together with p from $P(g,h)$ form a new primality type (μ,ψ,p) called a *finite primality type* which can be fully characterized. We can prove that the triple (μ,ψ,p) is a finite primality type if and only if either $0 = \mu = \psi = p$ or $1 \leq \mu \leq \psi \leq p < \infty$.

The end of this section is devoted to a third type of primality. We consider ways of adding an arbitrary language L to the definitions of P-, S- and F-prime solutions. More specifically, a solution is called F_L*-prime* if it cannot be divided in such a way that the prefix is a solution and the suffix belongs to L. So the only difference with the usual definition of F-prime solutions is that the part to be removed should now be an element of L. Consequently, $F(g,h) = F_{\Sigma^*}(g,h)$.

With this new definition it is especially interesting to study the effect of the different set-theoretic operations of languages on the corresponding L-prime solutions. For instance, it turns out that $F_{L_1 \cup L_2}(g,h) = F_{L_1}(g,h) \cap F_{L_2}(g,h)$ for all languages L_1 and L_2 but we can only say that $F_{L_1}(g,h) \cup F_{L_2}(g,h) \subseteq F_{L_1 \cap L_2}(g,h)$.

[Li4] also studies L-prime solutions in connection of some special language such as flat words or tense words. We say that a word is a k*-flatword* if and only if all its subwords of length r, $r \leq k+1$, contain each letter at most once. For instance, $w = abcbcad$ is a 1-flatword but not a 2-flatword. Now the flat prime solutions form an increasing hierarchy in every instance (g,h) where $\mathrm{card}\,(\Sigma) = n$:

$$\begin{array}{ccccccc}
P_{n-flat}(g,h) & \to & S_{n-flat}(g,h) & \to & F_{n-flat}(g,h) & \to & E(g,h) \\
\uparrow & & \uparrow & & \uparrow & & \\
\vdots & & \vdots & & \vdots & & \\
\uparrow & & \uparrow & & \uparrow & & \\
P_{2-flat}(g,h) & \to & S_{2-flat}(g,h) & \to & F_{2-flat}(g,h) & & \\
\uparrow & & \uparrow & & \uparrow & & \\
P_{1-flat}(g,h) & \to & S_{1-flat}(g,h) & \to & F_{1-flat}(g,h) & & \\
\uparrow & & \uparrow & & \uparrow & & \\
P(g,h) & \to & S(g,h) & \to & F(g,h) & &
\end{array}$$

The hierarchy can also collapse in some instances (g,h). For instance, if $|F_{k-flat}(g,h)| < \infty$ then $F_{k-flat}(g,h) = S_{k-flat}(g,h) = F(g,h) = S(g,h)$ and if $|F(g,h)| < \infty$ then either $S_{k-flat}(g,h) = F(g,h) = S(g,h)$ or $|S_{k-flat}(g,h)| = \infty$.

The hierarchy of the corresponding words, flat prime words, can also be characterized. It turns out that it is the same as for solutions except that every inclusion (an arrow) is proper in the previous figure and that no other relations exist; that is, the sets P_{k-flat} and $S_{(k-1)-flat}$ as well as S_{k-flat} and $F_{(k-1)-flat}$ are incomparable.

Tense words are in some sense opposite to flatwords. Now each subword of length k should contain all letters of the alphabet. For instance, *abca* is 3-tense and *ababa* is 2-tense. The tense prime words form again an increasing hierarchy. This time, however, we study separately F-, S- and P-hierarchies

$$F(g,h) \subseteq \ldots \subseteq F^2_{4-tense}(g,h) \subseteq F^2_{3-tense}(g,h) \subseteq F^2_{2-tense}(g,h) \subseteq E(g,h),$$
$$S(g,h) \subseteq \ldots \subseteq S^2_{4-tense}(g,h) \subseteq S^2_{3-tense}(g,h) \subseteq S^2_{2-tense}(g,h) \subseteq E(g,h),$$
$$P(g,h) \subseteq \ldots \subseteq P^2_{4-tense}(g,h) \subseteq P^2_{3-tense}(g,h) \subseteq P^2_{2-tense}(g,h) \subseteq E(g,h),$$

where the superscipt 2 indicates the binary alphabet.

The main results, concerning the hierarchies, in [Li4] settle the finiteness of the three hierarchies. S- and P-hierarchies turn out to be finite in every instance (g,h). This means that there always exists a k for which $S^2_{k-tense}(g,h) = S^2_{(k+1)-tense}(g,h) = S^2_{(k+2)-tense}(g,h)$ and so on.

For F-hierarchies this is also the case if g or h is injective. If both are periodic then the corresponding F-hierarchy is always infinite: for every k, $F^2_{k-tense}(g,h) - F^2_{(k+1)-tense}(g,h)$ is not empty.

5 Open problems. Conclusion

In addition to the conjectures we have presented throughout the text we would like to mention some other open problems. It seems that the binary alphabet makes an exception to many rules. Also the results are usually easier to achieve. The instances we can construct there are never very complicated which indicates that the binary case can possibly be completely settled.

But in larger alphabets lots of questions remain open. For instance, the nature of P- and S-words is beyond our present knowledge. Prime languages, on the other hand, lack a characterization in all alphabets.

We have emphasized the expressibility of equality sets: arbitrary computations can be expressed in terms of elements of equality sets. In view of the initial observations in [MSSY], it is an interesting research area to establish other explicit interconnections between primality of solutions and simplicity of computations.

References

[C] K. Culik II: A purely homomorphic characterization of recursively enumerable sets, *J. Assoc. Comput. Mach.* **26** (1979) 345–350.

[EKR] A. Ehrenfeucht, J. Karhumäki, G. Rozenberg: On binary equality sets and a solution to the test set conjecture in the binary case, *J. Algebra* **85** (1983) 76–85.

[HLM] T. Harju, M. Lipponen, A. Mateescu: Flatwords and Post Correspondence Problem, *Theoret. Comput. Sci.*, to appear.

[Hi] G. Higman: Ordering by divisibility in abstract algebras, *Proc. London Math. Soc.* **2** (1952) 326–336.

[LiIa] L. Kari: DNA Computers, Tomorrow's Reality, *EATCS Bull.* **59** (1996) 256–266.

[Li1] M. Lipponen: Primitive words and languages associated to PCP, *EATCS Bull.* **53** (1994) 217–226.

[Li2] M. Lipponen: Post Correspondence Problem: words possible as primitive solutions, *Proc. 22nd ICALP, Springer LNCS* **944** (1995) 63–74.

[Li3] M. Lipponen: On F-prime solutions of the Post Correspondence Problem. In J. Dassow, G. Rozenberg and A. Salomaa (ed.) *Developments in Language Theory II* (Magdeburg, 1995), World Scientific, Singapore (1996) 139–147.

[Li4] M. Lipponen: *On primitive solutions of the Post Correspondence Problem*, Turku Centre for Computer Science (TUCS) Dissertations, No 1 (1996). *http://www.tucs.abo.fi/publications/dissertations*

[LP] M. Lipponen, Gh. Păun: Strongly prime PCP words, *Discrete Appl. Math.* **63** (1995) 193–197.

[Ma] G.S. Makanin: The problem of solvability of equations in a free semigroup (in Russian), *Mat. Sb.* **103** No. 145 (1977) 148–236.

[MS1] A. Mateescu, A. Salomaa: PCP-prime words and primality types, *RAIRO Inform. Théor.* **27** (1993) 57–70.

[MS2] A. Mateescu, A. Salomaa: On simplest possible solutions for Post Correspondence Problems, *Acta Inform.* **30** (1993) 441–457.

[MS3] A. Mateescu, A. Salomaa: Formal languages, an introduction and a synopsis. In G. Rozenberg and A. Salomaa (ed.) *Handbook of Formal Languages, I-III*, Springer-Verlag, forthcoming.

[MSSY] A. Mateescu, A. Salomaa, K. Salomaa, Sheng Yu: P, NP and Post Correspondence Problem, *Inform. and Control* **121** (1995) 135–142.

[Post] E. Post: A variant of a recursively unsolvable problem, *Bull. Amer. Math. Soc.* **53** (1946) 264–268.

[RS] G. Rozenberg, A. Salomaa: *Cornerstones of Undecidability*, Prentice Hall (1994).

[Sa1] A. Salomaa: *Jewels of Formal Languages*, Computer Science Press (1981).

[Sa2] A. Salomaa: What Emil said about the Post Correspondence Problem. In G. Rozenberg and A. Salomaa (ed.) *Current Trends in Theoretical Computer Science*, World Scientific, Singapore (1993) 563–571.

[SSY] A. Salomaa, K. Salomaa, Sheng Yu: Primality types of instances of the Post Correspondence Problem, *EATCS Bull.* **44** (1991) 226–241.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**