J. HONKALA

A. SALOMAA

Caractérisation results about *L* codes

Informatique théorique et applications, tome 26, nº 3 (1992), p. 287-301

<http://www.numdam.org/item?id=ITA_1992__26_3_287_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 26, n° 3, 1992, p. 287 à 301)

CHARACTERIZATION RESULTS ABOUT *L* CODES (*)

by J. HONKALA and A. SALOMAA

Communicated by Jean BERSTEL

Abstract. – *The paper investigates interrelations between codes and* L *codes, presents characterization and decidability results for* L *codes of bounded delay, as well as discusses some related notions.*

Résumé. – *L'article étudie les interconnexions entre codes et* L *codes, présente des propriétés de caractérisation et de décidabilité pour les* L *codes à délai borné et discute des notions voisines.*

1. INTRODUCTION

L codes, introduced originally in [13], fit into the framework of generalizations and modifications of codes, an area that has been studied extensively during recent years. In order to make comparisons, it is useful to view a code as an injective morphism rather than a set of words, [2]. We return to this question in Section 2. *L* codes are obtained by applying a morphism

$$h: \quad \Sigma^* \to \Sigma^*$$

(not necessarily injective) in the "*L* way". This means applying h to the first letter of the argument word, h^2 to the second letter, h^3 to the third letter, and so on, and catenating the results. This gives rise to a mapping (that is a morphism only in special cases)

$$\bar{h}: \quad \Sigma^* \to \Sigma^*,$$

referred to as the *L associate* of h. The original h is called an *L code* iff $\bar{h}$ is injective. Every code is an *L* code but not vice versa, [13].

(*) Received November 1990, final version October 1991.
The Academy of Finland and Mathematics Department, University of Turku, 20500 Turku, Finland.

Apart from being a natural extension of the theory of codes, L codes are linked with the theory of number systems. This interconnection was observed already in [13]. The line of studies was continued in [7], [4] and [8]. For closely related work, *see* [1], [6] and [12]. Similarly as many classical cryptosystems can be viewed as codes, [17], also L codes are interesting from the point of view of cryptography, [18]. With this aspect in mind, we often use the term *plaintext* for the argument w and the term *cryptotext* for the value $h(w)$ or $\bar{h}(w)$.

Needless to say, L codes are closely linked with many problems dealing with L systems. We want to emphasize also that the related problems dealing with number systems have so far been investigated for unary morphisms only, that is, for morphisms whose range is generated by a single letter.

A brief outline of the contents of this paper follows. Section 2 contains the basic definitions, as well as explanations in case our definition deviates from the customary one. Technical lemmas needed later on will be established in Section 3. Section 4 presents the basics of L codes, in particular, the three different types of *bounded delay* L codes introduced in [16] and [14]. Two such types of bounded delay L codes are investigated in Sections 5 and 6: the family **S** of strongly bounded delay L codes and the family **M** of medium bounded delay L codes. More specifically, a characterization and a simple decision method are presented for **S** and a characterization for **M**. The last two sections are devoted to a further discussion and generalizations. In particular, the notion of an LL code is again closely linked with number systems. The paper is largely self-contained but [15] may be consulted if need arises.

Additional remark: This paper, together with the paper [10], constitute the full version of our ICALP-91 paper [11], where practically no proofs were given. More specifically, this paper is the full version of the "bounded delay" part of [11], whereas [10] is the full version of the "regularity of ambiguity" part. This paper contains also material not mentioned in [11], such as LL codes. The original version of this paper was written about half a year before [11].

2. MORPHISMS: DEFINITIONS

Consider a *nonerasing* morphism $h:\Sigma^* \to \Delta^*$, where Σ and Δ are finite alphabets. We want to emphasize that all morphisms discussed in this paper are nonerasing, that is, $h(a)\neq\lambda$ (the empty word) for every a in Σ. If h is injective it is referred to as a *code*. This definition was used, for instance, in [15]. For finite codes it is equivalent to the customary definition, [2], in the following sense. A morphism h is a code iff the set $\{h(a) \mid a \in \Sigma\}$ is a code, provided h is *non-identifying*, that is, $a \neq b$ implies $h(a) \neq h(b)$. In cryptographic terms h being a code means every "cryptotext" w' can be "decrypted" in at most one way, that is, there is at most one "plaintext" w such that $h(w) = w'$. The notation **C** will be used for the class of all codes.

Since morphisms will be iterated in the sequel, we consider only endomorphisms, that is, Δ is included in Σ.

For a positive integer k and a word w, we denote by $\mathrm{pref}_k(w)$ the prefix (initial subword) of w of length k. If w is shorter than k, then $\mathrm{pref}_k(w) = w$. The notation $\mathrm{first}(w)$ stands for the first letter of a nonempty word w. A morphism h is of *bounded delay* k if, for all words u and w, the equation

$$\mathrm{pref}_k(h(u)) = \mathrm{pref}_k(h(w))$$

implies the equation $\mathrm{first}(u) = \mathrm{first}(w)$. The morphism h is of *bounded delay* if it is of bounded delay k, for some k. The notation **B** will be used for the class of all morphisms of bounded delay. (Actually, our definitions concern bounded delay from left to right. This is the notion needed for L codes.)

When a code is viewed as a set, there are various definitions of bounded delay, [2, 3]. In particular, Bruyere considers three such definitions and shows that they are equivalent for finite sets. (This means that they all lead to the same collection of bounded delay sets, although the minimal value of k can be different under different definitions for the same set.) In particular, one of the definitions considered by her defines a set $X \subseteq \Sigma^+$ to be of bounded delay if, for some $k \geq 0$, the conditions

$$x_1 x y \in x_2 X^* \qquad \text{and} \qquad |x_1 x| > k,$$

where $x_1, x_2 \in X$, $x \in X^*$ and $y \in \Sigma^*$, imply the equation $x_1 = x_2$.

Assume now that h is non-identifying and the set $X = \{h(a) \mid a \in \Sigma\}$ is of bounded delay k. Define $k_1 = k + \max\{|h(a)| \mid a \text{ in } \Sigma\}$. Then the condition

$$\mathrm{pref}_{k_1}(h(au)) = \mathrm{pref}_{k_1}(h(bw)), \qquad a, b \in \Sigma; \quad u, w \in \Sigma^*,$$

implies that $h(a) = h(b)$ and, hence, $a = b$. (If $|h(au)| < k_1$, u and w can be replaced by longer words.) Conversely, assume that h is of bounded delay k. Then the conditions

$$h(a) h(x) y = h(b) h(x_1) \qquad \text{and} \qquad |h(a) h(x)| > k - 1,$$

where $a, b \in \Sigma$ and $x, x_1, y \in \Sigma^*$, imply $a=b$ and, hence, $h(a)=h(b)$. Thus, we have established the following

Observation: A non-identifying morphism h is of bounded delay iff the set $\{h(a) \mid a \in \Sigma\}$ is of bounded delay.

It is easy to prove that **B** is properly contained in **C**, [2]. We conclude this section with a few other definitions. A morphism h is a *prefix code* if there are no distinct letters a and b such that $h(a)$ is a prefix of $h(b)$. The class of all prefix codes is denoted by **P**. A morphism $h: \Sigma^* \to \Sigma^*$ is *elementary* if it is not *simplifiable*, that is, no alphabet Σ_1 smaller than Σ and no two morphisms $h_1 : \Sigma^* \to \Sigma_1^*$ and $h_2 : \Sigma_1^* \to \Sigma^*$ exist such that $h = h_2 h_1$.

For a word w, alph(w) denotes the minimal alphabet such that w is a word over this alphabet. For a morphism h and letter a, we say that a is *growing* with respect to h if, for every positive s, there is an i such that $|h^i(a)| > s$. Similarly, a is *stabile* with respect to h if $|h^i(a)| = 1$ for all i. If h is understood, we speak simply of growing and stabile letters. Clearly, there may be letters that are neither growing nor stabile.

3. MORPHISMS: BASIC LEMMAS

The rather diverse lemmas established in this section will be used in the sequel. The first lemma can be established in various ways using results concerning DOL systems with the axiom a. Our argument does not presuppose any knowledge concerning DOL systems.

LEMMA 1: *A letter a is growing with respect to h iff there are i and j, $j > i$, such that*

$$(*) \qquad |h^j(a)| > |h^i(a)| \qquad \textit{and} \qquad \mathrm{alph}(h^i(a)) = \mathrm{alph}(h^j(a)).$$

Given h and a, it is decidable whether or not a is growing with respect to h.

Proof: Recall that we consider only nonerasing morphisms. Conditions $(*)$ imply that

$$|h^{j+(n+1)(j-i)}(a)| > |h^{j+n(j-i)}(a)|, \qquad n \geqq 0,$$

which means that a is growing. Conversely, if the second condition in $(*)$ always implies that $|h^j(a)| = |h^i(a)|$ then, for any such i, $|h^n(a)| \leqq |h^i(a)|$ holds for all n and, thus, a is not growing. The second sentence of the lemma now follows because we only have to test whether or not the first condition in

(∗) holds for the smallest number j for which there is an i satisfying the second condition in (∗). □

The next two lemmas will be needed in Section 6. Lemma 2 is from [5] and Lemma 3 uses also ideas from [9].

LEMMA 2: *Assume that $h:\Sigma^* \to \Sigma^*$ is a morphism and $j \geqq \operatorname{card}(\Sigma)-1$. Then, for all words w and w' over Σ,*

$$(*) \qquad h^{j+1}(w)=h^{j+1}(w') \quad \textit{implies} \quad h^j(w)=h^j(w').$$

Proof: If h is injective (a condition always satisfied for $\operatorname{card}(\Sigma)=1$), then so are the powers of h. We proceed inductively, assuming that $h:\Sigma^* \to \Sigma^*$ is not injective and that the lemma holds for alphabets smaller than Σ. Since h is not injective, there are morphisms

$$h_1:\ \Sigma^* \to \Sigma_1^* \qquad \text{and} \qquad h_2:\ \Sigma_1^* \to \Sigma^*$$

such that $h=h_2 h_1$, $\operatorname{card}(\Sigma_1)<\operatorname{card}(\Sigma)$ and h_2 is injective. Assume that the first equation in (∗) holds. It can be written in the form

$$h_2(h_1 h_2)^j(h_1(w))=h_2(h_1 h_2)^j(h_1(w')).$$

By the injectivity of h_2 and by the inductive hypothesis, we infer successively

$$(h_1 h_2)^j(h_1(w))=(h_1 h_2)^j(h_1(w')), (h_1 h_2)^{j-1}(h_1(w))=(h_1 h_2)^{j-1}(h_1(w')).$$

By taking the h_2-images of both sides in the second equation, we obtain the second equation in (∗). □

LEMMA 3: *For all morphisms $h:\Sigma^* \to \Sigma^*$ and integers $n \geqq \operatorname{card}(\Sigma)-1$, there are morphisms g, p and q, of which p is elementary and g bounded delay, such that $h^{m+n}=gp^m q$, for all $m \geqq 0$.*

Proof: If h is elementary, we may choose $p=h$, $g=p^n$ and $q=$identity because (*see* [15]) elementary morphisms are of bounded delay and bounded delay morphisms are closed under composition. We proceed again by induction on $\operatorname{card}(\Sigma)$. Assume that $h:\Sigma^* \to \Sigma^*$ is not elementary and that the assertion holds for all alphabets smaller than Σ. Consequently, there is an alphabet Σ_1 smaller than Σ and morphisms

$$h_1:\ \Sigma^* \to \Sigma_1^* \qquad \text{and} \qquad h_2:\ \Sigma_1^* \to \Sigma^*$$

such that $h=h_2 h_1$ and h_2 is of bounded delay. We write h^{m+n} in the form

$$h^{m+n}=h_2(h_1 h_2)^{m+n-1} h_1.$$

Since $n-1 \geq \operatorname{card}(\Sigma_1)-1$, we obtain by the inductive hypothesis

$$(h_1 h_2)^{m+n-1} = g_1 p_1^m q_1,$$

where p_1 is elementary and g_1 of bounded delay. Consequently,

$$h^{m+n} = g p_1^m q,$$

where $q = q_1 h_1$ and $g = h_2 g_1$ is of bounded delay. □

Observe that Lemma 3 leads to an alternative proof of Lemma 2. The last two lemmas will be needed in Section 5.

LEMMA 4: *Assume that $h: \Sigma^* \to \Sigma^*$ is a code and a is in Σ. Then either* (i) *a is growing, or else* (ii) $|h^n(a)| = 1$ *for all n.*

Proof: Denote by Σ_1 the subset (possibly empty) of Σ consisting of letters satisfying (ii). Clearly, if a is in Σ_1 then so is $h(a)$. Furthermore, because h is a code, any two distinct letters a and b of Σ_1 satisfy $h(a) \neq h(b)$. This means that h permutes the set Σ_1, that is, $h(\Sigma_1) = \Sigma_1$. Moreover, for every a in Σ_1 and every integer $n \geq 0$, there is a (unique) letter a' of Σ_1 such that $h^n(a') = a$.

To prove the lemma, we assume the contrary: $a \in \Sigma$ satisfies neither (i) nor (ii), that is, there are integers $n \geq 1$ and $m > 1$ and letters $a_1, \ldots, a_m$ of Σ_1 such that $h^n(a) = a_1 \ldots a_m$. Consequently, there are letters $a_i' \in \Sigma_1$ such that $h^n(a_i') = a_i$, for $i = 1, \ldots, m$. But h^n is a code because codes are closed under composition. However,

$$h^n(a_1' \ldots a_m') = a_1 \ldots a_m = h^n(a),$$

a contradiction. □

LEMMA 5: *If h is a prefix code and a is growing then* first$(h(a))$ *is growing.*

Proof: The assertion is clearly true if $|h(a)| = 1$. Otherwise, $h(a) = bx$, for some letter b and nonempty word x. If b is not growing, the preceding proof shows that $h(c) = b$, for some c. But this contradicts the assumption of h being a prefix code. □

4. L CODES

Given a morphism $h:\Sigma^*\to\Sigma^*$, its *L associate* $\bar{h}$ is defined to be the mapping of Σ^* into Σ^* such that always

$$\bar{h}(a_1 a_2 \ldots a_n)=h(a_1)h^2(a_2)\ldots h^n(a_n),$$

where the a's are (not necessarily distinct) letters of Σ. (The empty word is mapped into itself by $\bar{h}$.) The morphism h is termed an *L code* if its L associate is injective, that is, there are no distinct words w_1 and w_2 such that

$$\bar{h}(w_1)=\bar{h}(w_2).$$

The class of L codes is denoted by **L**. Observe that it is not natural to define an L code as a set of words. Observe also that $\bar{h}$ is usually not a morphism. In fact, $\bar{h}$ is a morphism exactly in case h is idempotent. Moreover, $h\bar{h}\neq\bar{h}h$ with a few trivial exceptions. L codes are not closed under composition. The problem of deciding whether or not a given morphism is an L code was solved in [9] for morphisms giving rise to an empty set Σ_1, as defined in Lemma 4.

The notion of *bounded delay* for L codes was defined in [16] and [14]. The idea is the same as for codes: one has to read k letters of the cryptotext in order to determine the first plaintext letter. For codes the situation is unaltered, after the first plaintext letter a has been removed, as well as $h(a)$ from the cryptotext. The remainder of the cryptotext still equals $h(w)$, for some plaintext w. For L codes, the remainder of the cryptotext equals $h\bar{h}(w)$ rather than $\bar{h}(w)$. This means that we obtain different notions of bounded delay depending on whether we are interested in finding only the first plaintext letter (*weak* notion), or the first letter at each stage of decryption (*strong* or *medium strong* notion). The difference between the two latter notions emanates on the condition imposed on the bound of delay: is the bound constant (strong notion), or is it allowed to grow with the stage of decryption (medium strong notion). We are now ready for the formal definitions.

A morphism h is of *weakly bounded delay* $k\geq 1$ if, for all words u and w, the equation

$$\mathrm{pref}_k(\bar{h}(u))=\mathrm{pref}_k(\bar{h}(w))$$

implies the equation $\mathrm{first}(u)=\mathrm{first}(w)$. If for all $i\geq 0$ and all u and w, the equation

$$\mathrm{pref}_k(h^i\bar{h}(u))=\mathrm{pref}_k(h^i\bar{h}(w))$$

implies the equation $\text{first}(u) = \text{first}(w)$, then h is of *strongly bounded delay* k. In general, h is of weakly or strongly bounded delay if it is so for some k. The notations **W** and **S** are used for the corresponding classes of morphisms. Finally, h is of *medium bounded delay* if, for some recursive function f and all $i \geqq 0$, u and w, the equation

$$\text{pref}_{f(i)}(h^i \bar{h}(u)) = \text{pref}_{f(i)}(h^i \bar{h}(w))$$

implies the equation $\text{first}(u) = \text{first}(w)$. The notation **M** is used for the corresponding class.

Observe that we do not require h to be an L code in these definitions. The situation is analogous to that concerning ordinary codes. However, a morphism being in **B** implies that it is in **C**, [2], whereas **L** and **W** are incomparable. All inclusion relations between the classes introduced are presented in the following theorem. For the proof we refer to [14].

THEOREM 1: *The mutual inclusions between the families* **P**, **B**, **C**, **S**, **M**, **W** *and* **L** *are as follows:*

$$\begin{tikzcd}
& & & \mathbf{C} \arrow[r] & \mathbf{L} \\
\mathbf{S} \arrow[r] & \mathbf{P} \arrow[r] & \mathbf{B} \arrow[ur] \arrow[dr] & & \\
& & & \mathbf{M} \arrow[r] \arrow[uur] & \mathbf{W}
\end{tikzcd}$$

Here the arrow denotes strict inclusion, and two families are incomparable if they are not connected by a path.

5. THE FAMILY S

We now present a simple characterization for the family S. Some related questions will still be dealt with in Section 7.

THEOREM 2: *A morphism h is in* S *iff, for any distinct letters a and b,* $\text{first}(h(a)) \neq \text{first}(h(b))$.

Proof: Consider the "if"-part. The assumption means that there is a permutation π of the alphabet Σ such that, for all a,

$$\text{first}(h(a)) = \pi(a).$$

Consequently, for all $i \geqq 0$ and a,

$$\text{first}(h^i(a)) = \pi^i(a).$$

Therefore,

$$\text{pref}_1(h^i \bar{h}(a)) = \text{first}(h^{i+1}(a)) = \pi^{i+1}(a)$$

uniquely determines a, that is, h is of strongly bounded delay 1. In cryptographic terms: at the i-th stage of decryption the first letter in the remaining cryptotext uniquely determines the first letter of the still uncovered plaintext, but the decryption process depends on i.

Consider next the "only if"-part. Let h be in **S**. By Theorem 1, h is a prefix code. Proceeding indirectly, we assume that there are two distinct letters a and b such that $\text{first}(h(a)) = \text{first}(h(b))$. Since, h is a prefix code, we may write

$$h(a) = cxdy \qquad \text{and} \qquad h(b) = cxez,$$

where x, y, z are (possibly empty) words and c, d, e are letters such that $d \neq e$. By Lemma 4, a and b are growing letters. This implies, by Lemma 5, that also c is growing. Hence, for every k, there is an i such that

$$\text{pref}_k(h^i \bar{h}(a)) = \text{pref}_k(h^{i+1}(a)) = \text{pref}_k(h^{i+1}(b)),$$

which contradicts the assumption that h is in **S**. □

Theorem 2 gives a straightforward decision method for testing membership in **S**. We do not know any decision method for testing membership in **M** or **W**. Of course, for a fixed bound k, such a method is obvious.

6. THE FAMILY M

Medium bounded delay can be viewed in the theory of L codes as the most natural counterpart of bounded delay codes. It is natural to require that only a bounded amount of lookahead at each stage of the decryption process is needed. However, if the amount of lookahead remains the same throughout the process, the resulting notion is a very restricted one. This was seen in Section 5.

The drawback in the definition of **M** is that, in general, the construction of the sequence of values

$$f(i) = k_i, \qquad i = 0, 1, 2, \ldots,$$

seems to be an infinitary task. The purpose of this section is to show that, in fact, it suffices to construct the values only up to $\text{card}(\Sigma)-2$. [We assume that $\text{card}(\Sigma)\geqq 2$.] This construction already guarantees that the morphism is in **M**.

More specifically, we say that a morphism $h:\Sigma^*\to\Sigma^*$ is in the set $\mathbf{M}'$ if, for some $k>0$ and all i with $0\leqq i\leqq \text{card}(\Sigma)-2$, the equation

$$\text{pref}_k(h^i\bar{h}(u))=\text{pref}_k(h^i\bar{h}(w))$$

always implies the equation $\text{first}(u)=\text{first}(w)$.

Thus, we consider the sequence $f(i)=k_i$ only up to $\text{card}(\Sigma)-2$, and take the maximum of the resulting numbers.

THEOREM 3: $\mathbf{M}'=\mathbf{M}$.

Proof: The inclusion of the right side in the left side is obvious: if a recursive function f is associated with h as required in **M**, then a constant k as required in $\mathbf{M}'$ can be immediately found in the way indicated before the theorem.

To prove the reverse inclusion, we assume that h is in $\mathbf{M}'$. Assume that $i\geqq \text{card}(\Sigma)-1$. We give a method of computing $f(i)$ such that

$$\text{pref}_{f(i)}(h^i\bar{h}(u))=\text{pref}_{f(i)}(h^i\bar{h}(w))$$

always implies $\text{first}(u)=\text{first}(w)$. The value $f(i)$ depends on the constants involved; this will be explained more explicitly in the next section.

It will be more convenient to write u and w as products of letters. Thus, our basic equation assumes the form

$$(*)\quad \text{pref}_{f(i)}(h^{i+1}(a_1)h^{i+2}(a_2)\dots h^{i+s}(a_s))$$
$$=\text{pref}_{f(i)}(h^{i+1}(b_1)h^{i+2}(b_2)\dots h^{i+t}(b_t)).$$

By Lemma 2, we may exclude the case where the words appearing in $(*)$ are short. The following argument holds quite independently of the value $f(i)$. Assume that one of the words appearing in $(*)$ is shorter than $f(i)$. Then also the other must be so, and $(*)$ assumes the form

$$h^{i+1}(a_1)\dots h^{i+s}(a_s)=h^{i+1}(b_1)\dots h^{i+t}(b_t).$$

Hence, by Lemma 2,

$$h^i(a_1)\dots h^{i+s-1}(a_s)=h^i(b_1)\dots h^{i+t-1}(b_t).$$

If $i \geqq \text{card}(\Sigma)$, the argument can be repeated until the value $\text{card}(\Sigma)-1$ is reached. Hence, we may conclude that $a_1 = b_1$.

From now on we assume that the lengths of the words appearing in $(*)$ are at least $f(i)$. We denote $\text{card}(\Sigma)-1=n$ and rewrite $(*)$ according to Lemma 3:

$$(*)' \quad \text{pref}_{f(i)}(gp^{i+1-n}q(a_1)\ldots gp^{i+s-n}q(a_s)) = \text{pref}_{f(i)}(gp^{i+1-n}q(b_1)\ldots gp^{i+t-n}q(b_t)).$$

Observe that the length of $gp^{i+j-n}q(a)$, where a is a letter, is bounded by a constant. This means that $f(i)$ being large forces s and t to be large, too. This, in turn, gives enough lookahead (recall that g is of bounded delay) so that we can drop g from a long initial part of the words in $(*)'$. We may have to replace $f(i)$ by a smaller value $f'(i)$ because, after removing g, the common prefixes may be considerably shorter. Altogether, $(*)'$ assumes the form

$$(**) \quad \text{pref}_{f'(i)}(p^{i+1-n}q(a_1)\ldots p^{i+s'-n}q(a_{s'})) = \text{pref}_{f'(i)}(p^{i+1-n}q(b_1)\ldots p^{i+t'-n}q(b_{t'})),$$

where s' and t' are still large but smaller than s and t.

We now read the left side of $(**)$ as long as possible without exceeding $f'(i)$ in length. This gives rise to an equation

$$(**)' \quad p^{i+1-n}q(a_1)\ldots p^{i+s''-n}q(a_{s''}) = p^{i+1-n}q(b_1)\ldots p^{i+t''-n}q(b_{t''})z,$$

where we still assume s'' and t'' to be large. Observe that the "final mess" z is needed because the right side may be interrupted in the middle of morphic image. Observe also that we may obtain $(**)'$ directly from $(**)$ if the left side of $(**)$ is shorter than $f'(i)$.

Let k be the constant associated with h according to the definition of $\mathbf{M}'$. Since p is of bounded delay, also p^{i+1-n} is of bounded delay $d(i)$. We now view $(**)'$ in the form

$$p^{i+1-n}(\alpha_1\alpha_2\alpha_3\ldots) = p^{i+1-n}(\beta_1\beta_2\beta_3\ldots)z,$$

where the α's and β's are letters, and decrypt from the left, until on the right side the first letter of $p^k(q(b_{i+k}))$ has been reached. Since a full p^{i+1-n}-image remains on the left side, we obtain

$$p^{i+1-n+k}(q(b_{i+k}))\ldots p^{i+t''-n}(q(b_{t''}))z = p^{i+1-n}(z'),$$

for some z', and hence by $(**)'$

$$(***) \quad p^{i+1-n}q(a_1)\dots p^{i+s''-n}q(a_{s''}) = p^{i+1-n}q(b_1)\dots p^{i-n+k}q(b_k)p^{i+1-n}(z').$$

Here t'' has to be large enough to provide lookahead of size $d(i)$. Since p^{i+1-n} is injective, $(***)$ yields

$$q(a_1)\dots p^{s''-1}q(a_{s''}) = q(b_1)\dots p^{k-1}q(b_k)z'.$$

Taking first the g-images and then using the representation of h in terms of g, p and q, we obtain

$$\mathrm{pref}_k(h^n(a_1)h^{n+1}(a_2)\dots h^{n+s''-1}(a_{s''})) = \mathrm{pref}_k(h^n(b_1)h^{n+1}(b_2)\dots h^{n+k-1}(b_k)),$$

which implies $a_1 = b_1$. □

7. SCATTERED REMARKS

We now present some supplementary material to the two preceding sections. Let us first consider in more detail the computation of $f(i)$ in the preceding proof. The value $f(i)$ does not depend on $f(i-1)$. On the other hand, $f(i)$ depends on certain constants which, in turn, depend on i. Such constants are k, $d(i)$ and the delay d_g of g, as well as upper bounds of the form

$$B_H = \max\{\,|H(a)|\,|\,a \text{ in } \Sigma\},$$

where H is a composition of known morphisms, possibly depending on i.

The computation of $f(i)$ can be explained by going through the proof backwards. The number t'' has to be large enough to yield $(***)$. When we take into account the delay $d(i)$, we obtain the estimate

$$t'' > d(i) + k.$$

On the other hand, we must have $t > t'' + d_g$. In the estimate for $f(i)$, we must take the b-images as long as possible. Since $d(i)$ is linear (*see* [15]), our estimate for t is a linear function $e(i)$. Hence, we may choose

$$f(i) = B_1 + \dots + B_{e(i)},$$

where the B's are bounds of the form B_H. They may grow exponentially with i. (In fact, we are dealing here with DOL growth functions.) Thus we obtain an exponential expression for $f(i)$.

This shows that the class **M** remains unchanged if it is assumed that $f(i)$ is an exponential function. It is an open problem whether a better estimate can be obtained for $f(i)$.

It follows by Theorems 1 and 3 that the class **M**′ is contained in **L**. We now present a direct argument showing this, based on Lemma 2.

Assume that h is in **M**′ and that

$$\bar{h}(a_1 \ldots a_t) = \bar{h}(b_1 \ldots b_u).$$

Denoting again $n = \operatorname{card}(\Sigma) - 1$, we infer by the definition of **M**′ that $a_1 = b_1, \ldots, a_n = b_n$ and

$$h^n \bar{h}(a_{n+1} \ldots a_t) = h^n \bar{h}(b_{n+1} \ldots b_u).$$

Hence, by Lemma 2,

$$h^{n-1} \bar{h}(a_{n+1} \ldots a_t) = h^{n-1} \bar{h}(b_{n+1} \ldots b_u),$$

which implies $a_{n+1} = b_{n+1}$. Continuing in the same way, we conclude that $u = t$ and $a_i = b_i$ for $i = 1, \ldots, t$.

Finally, we outline a straightforward method for deciding membership in **S**. The method does not use Theorem 2 or Lemmas 4 and 5. The resulting algorithm is not as simple as the one obtained from Theorem 2. By Theorem 1, we may assume that the given morphism $h : \Sigma^* \to \Sigma^*$ is a prefix code.

For any pair (a, b) of distinct letters, there is a pair (c, d) of distinct letters such that

$$h(a) = wcx \qquad \text{and} \qquad h(b) = wdy,$$

where w, x, y are words (possibly empty). We denote

$$\operatorname{eq}(a, b) = w \qquad \text{and} \qquad \operatorname{diff}(a, b) = (c, d).$$

(It is possible that $\operatorname{diff}(a, b) = (a, b)$.) Let now $\operatorname{diffseq}(a, b)$ be the sequence (α_i, β_i), $i = 0, 1, \ldots$, with $(\alpha_0, \beta_0) = (a, b)$ and $(\alpha_{i+1}, \beta_{i+1}) = \operatorname{diff}(\alpha_i, \beta_i)$. Consider the smallest number n such that there is a number $m < n$ such that $(\alpha_m, \beta_m) = (\alpha_n, \beta_n)$. Define $w_i = \operatorname{eq}(\alpha_i, \beta_i)$, $i \geqq 0$, and the "deposit of period"

$$\operatorname{dep\,per}(a, b) = w_m \ldots w_{n-1}.$$

Then the verification of the following criterion is immediate.

The morphism h is in **S** iff, for every pair (a, b), $\mathrm{eq}(a, b)$ does not contain any growing letter and $\mathrm{dep\,per}(a, b)$ is empty.

The conditions are easily decidable, by Lemma 1.

8. LL CODES

The purpose of this section is to define a notion closely related to L codes and number systems.

A morphism $h: \Sigma^* \to \Sigma^*$ is a *lengthwise* L *code* or, briefly, *LL code* if all distinct words u and w satisfy $|\bar{h}(u)| \neq |\bar{h}(w)|$. The notation **LL** is used for the class of LL codes.

Assume that $\Sigma = \{a_1, \ldots, a_n\}$. The *growth matrix* M associated with h is an $n \times n$-matrix whose (i,j)-th entry equals the number of occurrences of a_j in $h(a_i)$. Let π_i be the i-th coordinate vector and η the n-dimensional column vector consisting of 1's. The morphism h is *unary* if there is a letter a_i such that $h(a_j)$ is a power of a_i, for all j.

THEOREM 4: *A unary morphism is an LL code iff it is an L code. The class* **LL** *is strictly contained in* **L**. *A morphism h is an LL code iff the function* $f: \Sigma^* \to N$ *defined by*

$$f(x) = (\sum_i \pi_i (\sum_j M^j))\, \eta$$

is an injection. Here the first sum is over the indices i such that a_i appears in x, and the second sum over the indices j such that a_i is the j-th letter in x.

Proof: The first sentence follows because in the unary case two words are different iff they differ in length. Similarly, **LL** is contained in **L** because difference in length, for any words, implies their difference. That the containment is strict is seen by considering the Fibonacci morphism: $h(a) = b$, $h(b) = ab$. It is, by Theorem 2, even in **S** but it is not in **LL** because $|\bar{h}(ab)| = |\bar{h}(ba)|$. The rest of the theorem follows from the definition of LL codes and the fact that $f(x) = |\bar{h}(x)|$. □

Observe that the condition given is not as such a decision method for membership in **LL**. Any decision method must also settle the uniqueness of representation in number systems, a problem solved in [7].

As regards the hierarchy of Theorem 1, **LL** is strictly contained in **L** but incomparable with all the other families in the hierarchy. This follows by the proof of Theorem 4 and the fact that **W** does not contain all unary L codes.

From the point of view of number systems, *LL* codes give the possibility of working with several bases at the same time. We hope to return to this area in a forthcoming paper.

REFERENCES

1. J. BERSTEL, Fibonacci Words—A Survey. *In* G. ROZENBERG and A. SALOMAA Ed. The Book of *L*, Springer-Verlag, Berlin, Heidelberg, New York, 1986, pp. 13-27.
2. J. BERSTEL and D. PERRIN, Theory of Codes, Academic Press, New York, 1985.
3. V. BRUYERE, Codes prefixes. Codes a delai de déchiffrage borne. *Nouvelle thèse*, Université de Mons, 1989.
4. K. CULIK II and A. SALOMAA, Ambiguity and Decision Problems Concerning Number Systems, *Inform. and Control*, 1983, *56*, pp. 139-153.
5. A. EHRENFEUCHT and G. ROZENBERG, Simplifications of Homomorphisms, *Inform. and Control*, 1978, *38*, pp. 298-309.
6. C. FROUGNY, Linear Numeration Systems of Order Two, *Inform. and Comput.*, 1988, *77*, pp. 233-259.
7. J. HONKALA, Unique Representation in Number Systems and *L* Codes, *Discrete Appl. Math.*, 1982, *4*, pp. 229-232.
8. J. HONKALA, Bases and Ambiguity of Number Systems, *Theoret. Comput. Sci.*, 1984, *31*, pp. 61-71.
9. J. HONKALA, It is Decidable Whether or Not a Permutation-Free Morphism is an *L* Code, *Int. J. Comput. Math.*, 1987, *22*, pp. 1-11.
10. J. HONKALA, Regularity Properties of *L* Ambiguities of Morphisms, "In the Footsteps of *L*", *Springer-Verlag* (to appear).
11. J. HONKALA and A. SALOMAA, *L*-Morphisms: Bounded Delay and Regularity of Ambiguity, *Springer Lecture Notes in Comput. Sci.*, 1991, *510*, pp. 566-574.
12. A. DE LUCA and A. RESTIVO, Star-free Sets of Integers, *Theoret. Comput. Sci.*, 1986, *43*, pp. 265-275.
13. H. MAURER, A. SALOMAA and D. WOOD, *L* Codes and Number Systems, *Theoret. Comput. Sci.*, 1983, *22*, pp. 331-346.
14. H. MAURER, A. SALOMAA and D. WOOD, Bounded delay *L* codes, IIG Technical Report 1990, *Theoret. Comput. Sci.*, 1991, *84*, pp. 265-279.
15. A. SALOMAA, Jewels of Formal Language Theory, Computer Science Press, Rockville, 1981.
16. A. SALOMAA, *L* codes: variations on a theme by MSW. *In* Ten Years IIG, IIG Report, 1988, *260*, p. 218.
17. A. SALOMAA, Public-Key Cryptography, *Springer-Verlag*, Berlin, Heidelberg, New York, 1990.
18. A. SALOMAA, Cryptographic properties of *L* codes, in preparation.