f-Words and Binary Solid Codes[1]

Stavros Konstantinidis[†] and Joshua Young[†]

[†]Department of Mathematics and Computing Science
Saint Mary's University
Halifax, Nova Scotia, B3H 3C3 Canada
s.konstantinidis@smu.ca, jyo04@hotmail.com

Abstract. Given any unbounded and non-decreasing sequence f of positive integers, we define an infinite set of binary words, called f-words, which constitute an overlap-free language. We investigate some properties of this language and then use these properties to define new classes of finite and infinite binary solid codes – solid codes have the strongest synchronization and error-delimiting capabilities in the hierarchies of codes. The finite class improves on an earlier construction of solid codes in terms of average word length (or information rate), without sacrificing their encoding complexity. The infinite class concerns maximal solid codes and builds on an earlier work on maximal binary solid codes. This work constitutes another step towards a systematic structural characterization of binary maximal solid codes.

Key words: construction, encoding complexity, maximal, overlap-free language, solid code, word.

1 Introduction

From a language theoretic point of view, a solid code is an infix code (no codeword is contained in another codeword) and an overlap-free language (no proper prefix of a codeword is also a proper suffix of some codeword, unless it is empty). Solid codes constitute a proper subclass of comma-free codes, providing thus synchronization of messages without delay. In addition, they possess the following remarkable property: if a message over a solid code undergoes any type of errors, then the codewords containing no errors can be identified and decoded correctly without delay. We refer the reader to [4] for a relevant discussion and further references on these codes, as well as to [3], [8], [1], [6], [10], [13], [2] for more recent results on these objects.

The general problem of constructing "good" solid codes is of central importance in our context. Some of the criteria used in the literature for evaluating the "goodness" of codes are the following: maximality, information ratio (or average word length), encoding/decoding complexity, and error-detectability. In addition, there have been systematic efforts to characterize explicitly, or generate algorithmically, all possible solid codes of certain types [3], [8].

In [3], and then in [6], the authors presented constructions of (binary) solid codes that are defined via pairs of functions h and g and have words of the form

$$a^{h(\dot{z})}b^{z_0}a^{z_1}\cdots b^{z_{2k-2}}a^{z_{2k-1}}b^{g(\dot{z})}, \tag{1}$$

such that $\dot{z}$ is the tuple $(z_0,\ldots,z_{2k-1})$ of the positive integers appearing in the *runs* $b^{z_0}, a^{z_1},\ldots$ of the above word – see the next section for definitions of technical terms. In [3] the authors showed a complete structural characterization of *all* maximal solid codes that are subsets of $a^+b^+a^+b^+$, by defining the exact requirements for the functions h and g. In [6], using again the approach of word runs, a class of finite solid codes was shown that has an asymptotically optimal information ratio

[1]Research supported by a Discovery Research Grant of NSERC, Canada.

and very simple linear encoding/decoding complexity. Moreover, some of these codes have certain good error-detecting capabilities. In [9], [10], the author presented a simple class of fixed-length solid codes that appear to be the best possible in terms of information ratio.

In this paper, we continue the systematic investigation of binary solid codes that are defined in terms of the runs of the words involved, in view of the fact, [3], that *every* binary solid code can be defined via a pair of functions h and g as shown in (1). It turns out that an explicit characterization of all maximal solid codes that are subsets of $(a^+b^+)^k$, for some integer $k \geq 2$, is quite difficult to describe. Our investigations have led to the discovery of f-words. These are words of the form shown in the expression (1), where h and g are defined explicitly via the choice of any non-decreasing and unbounded sequence f of positive integers. In particular, the sets of all f-words of the form $a^+b^+a^+b^+$ are exactly all the maximal solid codes defined in [3]. The f-words can also be used to improve the information rate of the finite solid codes defined in [6], without sacrificing their encoding/decoding complexity.

This paper is organized as follows. The next section contains the basic notions and notation used throughout the paper. Section 3 contains the definition of f-words and shows some basic properties of these words. In particular, the set of these words is an infinite overlap-free language. Section 4 uses f-words to improve the construction of the finite solid codes in [6], and Section 5 defines, again via f-words, new classes of infinite solid codes, including a class of infinite maximal solid codes that are subsets of

$$a^+b^+a^+b^+a^+b^+ \cup a^+b^+a^+b^+a^+b^+a^+b^+.$$

Finally, Section 6 contains a few concluding remarks.

2 Basic Notions and Notation

The symbol $\mathbb{N}$ denotes the set of positive integers. An *alphabet* is any finite and nonempty set of elements, which we call symbols or letters. We shall use the symbol Σ for the binary alphabet

$$\Sigma = \{a, b\}.$$

As usual, we use the notation Σ^* for the set of all words over the alphabet Σ, including the empty word λ, and Σ^+ for $\Sigma^* - \{\lambda\}$. The length of a word u is denoted as $|u|$ and is defined to be the number of symbols occurring in u. Any set of words is called a *language*. The concatenation of two words u and v is written as uv. This notation is extended naturally to more than two words: $u_1u_2\cdots u_n$ is the concatenation of the words $u_1, u_2, \ldots, u_n$. When all the u_i's are the same, say equal to v, we write v^n for their concatenation. Obviously $u\lambda = \lambda u = u$, for all words u. The notation for word concatenation is extended naturally to languages. For example, if K and L are languages, then K^2L is the language of all words of the form v_1v_2v such that $v_1, v_2 \in K$ and $v \in L$.

If a word u is of the form xy, then x is called a *prefix* of u and y is called a *suffix* of u. If x is not equal to u then it is called a proper prefix – proper suffixes are defined analogously. If u is of the form xyz, then y is an *infix* of u – obviously prefixes and suffixes are special infixes of a word. A *run* of u is an infix of u of the form σ^n, for some $\sigma \in \Sigma$, such that u can be written as $x\sigma^n y$, and x does not end with σ and y does not start with σ.

The expression (S_n) denotes a sequence $S_1, S_2, S_3, \ldots$ of sets. Let (K_n) be a sequence of finite languages over $\{a, b\}$ such that each K_n contains at least 2^n words. We say that (K_n) has *linear time encoding complexity* if there is a 1-1 function from $\{a, b\}^+$ to $\cup_{n=1}^{\infty} K_n$ that maps every word of length n to a unique word in K_n such that the function can be computed in time $O(n)$. For

example, for any word $v \in \{a,b\}^+$, let $p_v \in \{a,b\}$ be the even parity bit for v, that is, $p_v = b$ if and only if v has an even number of a's. Let

$$P_n = \{vp_v \mid v \in \{a,b\}^n\}$$

– this is the even parity code of length $n+1$. Then (P_n) has linear time encoding complexity, as each word v of length n can be encoded as vp_v by reading v once and counting the number of a's in v.

Two nonempty words have an *overlap* z, if z is a nonempty word such that z is a proper prefix of one of the words and a proper suffix of the other. For example aa is an overlap of the words $aaabb$ and $ababaa$. A language L is an *overlap-free language* if no two words of L have an overlap. Here we use the definition of [4] for an overlap-free language. We note that this term is defined slightly differently in [12]. A language L is an *infix code*, if no L-word is a proper infix of another L-word. A language is called a *solid code* (or, code without overlaps) if it is both an infix code and an overlap-free language. If $\mathcal{C}$ is any class of languages/codes (infix, overlap-free, etc.), then we say that a language/code L is a *maximal* $\mathcal{C}$ language/code, if for any word w outside of L, namely $w \in \Sigma^* - L$, the set $L \cup \{w\}$ is not in the class $\mathcal{C}$.

A *tuple* is any finite sequence of positive integers written as

$$\dot{z} = (z_0, \ldots, z_{k-1}). \tag{2}$$

We use the dot notation $\dot{z}$ for tuples. The empty tuple is ( ). The *length* of $\dot{z}$ is the number of components in $\dot{z}$ – this is equal to k in the tuple shown in (2). The set of all tuples of length k is denoted as $\mathbb{N}^k$ and the set of all tuples as $\mathbb{N}^*$. The *size* of the tuple $\dot{z}$ is $\sum \dot{z} = \sum_{i=0}^{k-1} z_i$. We also define $\max \dot{z} = \max\{z_i \mid i = 0, \ldots, k-1\}$.

Consider any unbounded and non-decreasing function $f : \mathbb{N} \to \mathbb{N}$ – here, non-decreasing means $f(i) \leq f(i+1)$, whereas increasing means $f(i) < f(i+1)$, for all $i \in \mathbb{N}$. The *near inverse* of f is the function $\hat{f} : \mathbb{N} \to \mathbb{N}$ such that

$$\hat{f}(j) = \min\{i \in \mathbb{N} \mid f(i) > j\}.$$

The term near-inverse arises from the fact that, if φ is any real increasing function with $\varphi(i) = f(i)$ for all $i \in \mathbb{N}$, then

$$\hat{f}(j) = 1 + \lfloor \varphi^{-1}(j) \rfloor.$$

In [5], it is shown that, like f, the near-inverse $\hat{f}$ is unbounded and non-decreasing. A basic property of near-inverses is

$$\text{For all } i, j \in \mathbb{N}: \quad f(i) > j \text{ if and only if } i \geq \hat{f}(j). \tag{3}$$

Example 1 *When $f(i) = 2i$ it is easy to see that $\hat{f}(j) = 1 + \lfloor j/2 \rfloor$. In the sequel, we shall use the identity function $\delta(i) = i$ whose near-inverse is $\hat{\delta}(j) = 1 + j$.*

3 f-Words

We define a certain type of words, called f-words, with the aim that any two such words are overlap free. The definition of these words is inspired from the constructions of solid codes in [3] and [6]. In particular, in [3] it is noted that every binary solid code that is a subset of $a\Sigma^*b$ is defined using a set of tuples $\mathbb{T}$ of even length, and two functions $h, g : \mathbb{T} \to \mathbb{N}$ such that each word of the code is

of the form shown in (1), where $\dot{z}$ is a tuple in $\mathbb{T}$ of some length $2k$. In [6], the set $\mathbb{T}$ consists of all tuples $\dot{z}$ such that $\sum \dot{z} = n$, for some fixed word length n, and

$$h(\dot{z}) = \max\{f(z_{2i}) : i = 0, \ldots, k-1\} \text{ and } g(\dot{z}) = \max\{\hat{f}(z_{2i-1}) : i = 1, \ldots, k\},$$

where $f : \mathbb{N} \to \mathbb{N}$ is any non-decreasing and unbounded function. Thus, $h(\dot{z})$ is the f-value on the length of the longest b-run, and $g(\dot{z})$ is the $\hat{f}$-value on the length of the longest a-run in $b^{z_0}a^{z_1}\cdots b^{z_{2k-2}}a^{z_{2k-1}}$.

For reasons of notational convenience we view a tuple $\dot{z}$ of even length as consisting of two interleaved tuples $\dot{x}$ and $\dot{y}$ of the same length k, where we use the components of the second one in reverse order. More specifically, for any two tuples $\dot{x}, \dot{y}$ of some length k, we define the word

$$\mathrm{wd}(\dot{x}, \dot{y}) = b^{x_0}a^{y_{k-1}}b^{x_1}\cdots a^{y_{k-i}}b^{x_i}\cdots a^{y_1}b^{x_{k-1}}a^{y_0},$$

where we assume that $\mathrm{wd}(\dot{x}, \dot{y})$ is empty if $k = 0$.

In the sequel we assume that $f : \mathbb{N} \to \mathbb{N}$ is any non-decreasing and unbounded function. We use this function to define, for any tuples $\dot{x}$ and $\dot{y}$, the quantities $f_{\dot{x},\dot{y}} = h(\dot{x}, \dot{y})$ and $\hat{f}_{\dot{y},\dot{x}} = g(\dot{x}, \dot{y})$ that will be used to make overlap-free words as shown in the expression (1). One possibility is to use $f_{\dot{x},\dot{y}} = f(\max \dot{x})$ and $\hat{f}_{\dot{y},\dot{x}} = \hat{f}(\max \dot{y})$, as in [6]. However, we can do better than that. Instead of giving immediately the definitions for $f_{\dot{x},\dot{y}}$ and $\hat{f}_{\dot{y},\dot{x}}$, we provide a couple of examples that motivate the choice of our definitions. Any two (possibly equal) f-words will be of the form

$$a^{f_{\dot{x},\dot{y}}}\mathrm{wd}(\dot{x}, \dot{y})b^{\hat{f}_{\dot{y},\dot{x}}}, \quad a^{f_{\dot{s},\dot{t}}}\mathrm{wd}(\dot{s}, \dot{t})b^{\hat{f}_{\dot{t},\dot{s}}}. \tag{4}$$

An overlap of these words has the form $a^{f_{\dot{x},\dot{y}}}b^{\hat{f}_{\dot{t},\dot{s}}}$ (we call this case "$i = -1$"), or

$$a^{f_{\dot{x},\dot{y}}}b^{x_0}(a^{y_{k-1}}b^{x_1})\cdots(a^{y_{k-i}}b^{x_i})a^{y_{k-i-1}}b^{\hat{f}_{\dot{t},\dot{s}}} = a^{f_{\dot{x},\dot{y}}}b^{s_{l-i-1}}(a^{t_i}b^{s_{l-i}})\cdots(a^{t_1}b^{s_{l-1}})a^{t_0}b^{\hat{f}_{\dot{t},\dot{s}}}, \tag{5}$$

where $\hat{f}_{\dot{t},\dot{s}} \leq x_{i+1}$ and $f_{\dot{x},\dot{y}} \leq t_{i+1}$, and for $i \geq 0$ the tuples match as follows

$$(x_0, \ldots, x_i) = (s_{l-i-1}, \ldots, s_{l-1}) \text{ and } (y_{k-1}, \ldots, y_{k-i-1}) = (t_i, \ldots, t_0).$$

For the case "$i = -1$", the overlap is prevented when we *require* that always $f(x_0) \leq f_{\dot{x},\dot{y}}$ and $\hat{f}(t_0) \leq \hat{f}_{\dot{t},\dot{s}}$. Indeed, in this case, we get $f(x_0) \leq t_0$ and $\hat{f}(t_0) \leq x_0$, and by Property (3) of near-inverses, $x_0 < \hat{f}(t_0)$; a contradiction.

For the case "$i = 0$", the overlap in (5) is of the form

$$a^{f_{\dot{x},\dot{y}}}b^{x_0}a^{y_{k-1}}b^{\hat{f}_{\dot{t},\dot{s}}} = a^{f_{\dot{x},\dot{y}}}b^{s_{l-1}}a^{t_0}b^{\hat{f}_{\dot{t},\dot{s}}}$$

such that $f_{\dot{x},\dot{y}} \leq t_1$ and $\hat{f}_{\dot{t},\dot{s}} \leq x_1$, and $x_0 = s_{l-1}$, $t_0 = y_{k-1}$. With these facts, the requirement $\hat{f}(t_0) \leq \hat{f}_{\dot{t},\dot{s}}$ yields $\hat{f}(y_{k-1}) \leq x_1$, which implies $y_{k-1} < f(x_1)$ – see Property (3). By symmetry, also $\hat{f}(t_1) > s_{l-1}$ holds. Now in order to prevent the above overlap (case "$i = 0$") from happening, we further *require* that $f_{\dot{x},\dot{y}} \geq f(x_1)$ if $y_{k-1} < f(x_1)$ and, analogously, $\hat{f}_{\dot{t},\dot{s}} \geq \hat{f}(t_1)$ if $\hat{f}(t_1) > s_{l-1}$. Indeed, as both f and $\hat{f}$ are non-decreasing, $f_{\dot{x},\dot{y}} \geq f(x_1)$ implies $f_{\dot{x},\dot{y}} \geq f(\hat{f}_{\dot{t},\dot{s}})$, and $\hat{f}_{\dot{t},\dot{s}} \geq \hat{f}(t_1)$ implies $\hat{f}_{\dot{t},\dot{s}} \geq \hat{f}(f_{\dot{x},\dot{y}})$. These two implied facts lead to a contradiction in view of Property (3) of near inverses.

For longer overlaps of the form shown in (5) the requirements for $f_{\dot{x},\dot{y}}$ and $\hat{f}_{\dot{t},\dot{s}}$ become more complex. Next we give the definition for these quantities. It is based on the following sets of indices

$$I_f(\dot{x}, \dot{y}) = \{0\} \cup \{i \mid 0 < i < k,\ \max(y_{k-1}, \ldots, y_{k-i}) < f(x_i)\}, \tag{6}$$

$$I_{\hat{f}}(\dot{y}, \dot{x}) = \{0\} \cup \{j \mid 0 < j < k,\ \hat{f}(y_j) > \max(x_{k-j}, \ldots, x_{k-1})\}. \tag{7}$$

Example 2 *Using the function $f(i) = 2i$, with $\hat{f}(j) = 1 + \lfloor j/2 \rfloor$, and the word*

$$\mathrm{wd}(\dot{x}, \dot{y}) = ba^3bab^2a^4b^2a^2$$

we see that $I_f(\dot{x}, \dot{y}) = \{0, 2\}$ and $I_{\hat{f}}(\dot{y}, \dot{x}) = \{0, 1\}$.

Definition 1 *For any non-decreasing and unbounded function $f : \mathbb{N} \to \mathbb{N}$, and tuples $\dot{x}$ and $\dot{y}$ of the same length k,*

$$f_{\dot{x},\dot{y}} = \max\{f(x_i) \mid i \in I_f(\dot{x}, \dot{y})\} \quad (8)$$

$$\hat{f}_{\dot{y},\dot{x}} = \max\{\hat{f}(y_j) \mid j \in I_{\hat{f}}(\dot{y}, \dot{x})\} \quad (9)$$

An f-word is any word of the form

$$\mathrm{wd}_f(\dot{x}, \dot{y}) = a^{f_{\dot{x},\dot{y}}}\, \mathrm{wd}(\dot{x}, \dot{y})\, b^{\hat{f}_{\dot{y},\dot{x}}} = a^{f_{\dot{x},\dot{y}}}\, b^{x_0} a^{y_{k-1}} b^{x_1} \cdots a^{y_{k-i}} b^{x_i} \cdots a^{y_1} b^{x_{k-1}} a^{y_0}\, b^{\hat{f}_{\dot{y},\dot{x}}}.$$

The language of all f-words is

$$L_f = \{\, \mathrm{wd}_f(\dot{x}, \dot{y}) \mid \dot{x}, \dot{y} \in \mathbb{N}^k, \ \textit{for some } k > 0 \,\}.$$

Using again the above word $\mathrm{wd}(\dot{x}, \dot{y}) = ba^3bab^2a^4b^2a^2$, we see that $\mathrm{wd}_f(\dot{x}, \dot{y}) = a^4ba^3bab^2a^4b^2a^2b^3$.

Lemma 1 *Let w be any nonempty word having a^t as a longest run of a's and b^s as a longest run of b's, with $s, t > 0$.*

1. *If w is a proper prefix of any f-word then $\hat{f}(t) > s$.*
2. *If w is a proper suffix of any f-word then $f(s) > t$.*

Proof. We only prove the first statement, as the second one follows by symmetry. Let $\mathrm{wd}_f(\dot{x}, \dot{y})$ be any f-word containing w as a proper prefix such that the tuples $\dot{x}$ and $\dot{y}$ are of length k, for some $k \in \mathbb{N}$. We argue by contradiction, assuming that $\hat{f}(t) \leq s$ and, therefore, $t < f(s)$. As $a^{f_{\dot{x},\dot{y}}}$ is a run in w, we have $f_{\dot{x},\dot{y}} \leq t$. Hence, $f_{\dot{x},\dot{y}} < f(s)$. Thus, so far we have obtained

$$\hat{f}(t) \leq s, \quad f_{\dot{x},\dot{y}} \leq t, \quad f_{\dot{x},\dot{y}} < f(s). \quad (10)$$

Let w be of the form $w_1 b^s w_2$, for some words w_1 and w_2, such that w_1 contains no run b^s. We continue by distinguishing three cases about the structure of $w_1 b^s$.

In the first case, $w_1 b^s = a^{f_{\dot{x},\dot{y}}} b^s$. Then, by the definition of $f_{\dot{x},\dot{y}}$, we have that $f_{\dot{x},\dot{y}} \geq f(s)$, which contradicts (10).

In the second case, $w_1 b^s$ is equal to

$$a^{f_{\dot{x},\dot{y}}} b^{x_0} a^{y_{k-1}} b^{x_1} \cdots a^{y_{k-i}} b^s$$

for some i with $0 < i \leq k - 1$ and $x_i \geq s$. As f is non-decreasing, $f(x_i) \geq f(s) > t$, so $f(x_i)$ is greater than all the y_j's appearing in w_1. This implies $i \in I_f(\dot{x}, \dot{y})$ and, therefore, $f_{\dot{x},\dot{y}} \geq f(x_i) > t$, which contradicts (10).

In the third case, w_2 is empty, $w = w_1 b^s$ and w ends in the last run $b^{\hat{f}_{\dot{y},\dot{x}}}$ of $\mathrm{wd}_f(\dot{x}, \dot{y})$, and b^s is a proper prefix of $b^{\hat{f}_{\dot{y},\dot{x}}}$. Hence, $\hat{f}_{\dot{y},\dot{x}} > s$. As $\hat{f}_{\dot{y},\dot{x}} = \hat{f}(y_j)$ for some index j, and a^{y_j} is a run of w, we have that $\hat{f}(t) \geq \hat{f}_{\dot{y},\dot{x}} > s$, which again contradicts (10). ■

Theorem 1 *For every non-decreasing and unbounded function f, the language L_f of all f-words is an overlap-free language.*

Proof. If a nonempty w is a proper prefix and proper suffix of two f-words, then a contradiction arises using the above lemma and the properties of near-inverses. Hence, L_f is overlap-free. ■

A natural question that arises here is whether L_f is a maximal overlap-free language. As shown next, this is not the case. However, every word w outside of L_f that has no overlap with any L_f-word must be a proper prefix, or a proper suffix of some L_f-word. For example, for $f(i) = 2i$, with $\hat{f}(j) = 1 + \lfloor j/2 \rfloor$, we have the following

$$w = a^6ba^2b^3 \notin L_f,\ \ a^2ba^2b^2 \in L_f,\ \ wa^4b^2a^3b^3 \in L_f.$$

Obviously, the word w itself is overlap-free. If a proper nonempty prefix, say v, of w is a proper suffix of some L_f-word, then the word v is also a proper prefix of the L_f-word $wa^4b^2a^3b^3$, which is impossible. Similarly, if a proper nonempty suffix u of w is a proper prefix of some f-word $a^{f_{\dot{x},\dot{y}}}\mathrm{wd}(\dot{x},\dot{y})b^{\hat{f}_{\dot{y},\dot{x}}}$ then

$$u \in \{a^iba^2b^3, a^jb^3 \mid i = 1, \ldots, 5;\ j = 1, 2\},$$

which implies that $a^{f_{\dot{x},\dot{y}}} = a^i$, or $a^{f_{\dot{x},\dot{y}}} = a^j$, as shown above and, therefore, $f_{\dot{x},\dot{y}} < 6$. On the other hand, by the definition of $f_{\dot{x},\dot{y}}$ and the fact that u is a prefix of $a^{f_{\dot{x},\dot{y}}}\mathrm{wd}(\dot{x},\dot{y})b^{\hat{f}_{\dot{y},\dot{x}}}$, it must be that $f_{\dot{x},\dot{y}} \geq f(3) = 6$, which is again impossible. Hence, $L_f \cup \{w\}$ is an overlap-free language.

Proposition 1 *Let w be any word in $\Sigma^* - L_f$. Then w has an overlap with some word in L_f, or w is a proper prefix, or proper suffix, of some word in L_f.*

Proof. Consider any word w in $\Sigma^* - L_f$, and assume for the sake of contradiction that w has no overlap with any word in L_f, and that w is neither a prefix nor a suffix of any word in L_f.

If w starts with b or ends with a, then w has an overlap with every word in L_f; a contradiction. If w is of the form a^tb^s, then we consider the word $u = a^{f(s)}b^sa^tb^{\hat{f}(t)} \in L_f$, and we observe that, as $a^tb^{\hat{f}(t)}$ is not a proper prefix of w and w is not a proper prefix of u, we must have $\hat{f}(t) > s$. Similarly, we must have $f(s) > t$, which leads to a contradiction by the properties of near-inverses.

Now suppose that w is of the form

$$w = a^{y_m}b^{x_0}a^{y_{m-1}}b^{x_1}\cdots b^{x_{m-1}}a^{y_0}b^{x_m},$$

for some $m \geq 1$. We show that, for all $i = 0, \ldots, m$,

$$x_i < \max\{\hat{f}(y_m), \ldots, \hat{f}(y_{m-i})\}. \tag{11}$$

Indeed, assume that there is an index i such that $x_i \geq \hat{f}(y_m), \ldots, \hat{f}(y_{m-i})$, and consider the word

$$u = a^{f_{\dot{s},i}}b^{x_0}a^{y_m}b^{x_0}\cdots a^{y_{m-i+1}}b^{x_{i-1}}a^{y_{m-i}}b^{\hat{f}_{\dot{t},\dot{s}}} \in L_f,$$

where $\dot{s} = (x_0, x_0, \ldots, x_{i-1})$ and $\dot{t} = (y_m, \ldots, y_{m-i})$. Then the word

$$z = a^{y_m}b^{x_0}\cdots a^{y_{m-i+1}}b^{x_{i-1}}a^{y_{m-i}}b^{\hat{f}_{\dot{t},\dot{s}}}$$

is a proper suffix of u. If $i < m$, or $i = m$ and $\hat{f}_{\dot{t},\dot{s}} < x_m$, then z is an overlap of w and u, which is a contradiction. If $i = m$ and $\hat{f}_{\dot{t},\dot{s}} = x_m$, then w is a proper suffix of u, which is again a contradiction. Hence, (11) holds. By symmetry, we also get that, for all $j = 0, \ldots, m$,

$$y_{m-j} < \max\{f(x_j), \ldots, f(x_m)\}. \tag{12}$$

Now it follows that $\max \dot{x} < \hat{f}(y_{m-j})$, for some index j, which implies $f(\max \dot{x}) \leq y_{m-j}$. Also, by (12), we have $y_{m-j} < f(\max \dot{x})$, which is a contradiction, as required. ■

We close this section by proving an interesting property of an f-word.

Proposition 2 *For every f-word $\mathrm{wd}_f(\dot{x}, \dot{y})$ we have that*

$$(f_{\dot{x},\dot{y}} = f(\max \dot{x}) > \max \dot{y} \;\; \textit{and} \;\; \hat{f}_{\dot{y},\dot{x}} \leq \max \dot{x}) \;\; \textit{OR} \;\; (\hat{f}_{\dot{y},\dot{x}} = \hat{f}(\max \dot{y}) > \max \dot{x} \;\; \textit{and} \;\; f_{\dot{x},\dot{y}} \leq \max \dot{y})$$

Proof. The proof relies on a sequence of three claims. The first claim is

$$(F1): \quad (f_{\dot{x},\dot{y}} > \max \dot{y}) \;\; \text{OR} \;\; (\hat{f}_{\dot{y},\dot{x}} > \max \dot{x}),$$

which we show using contradiction. So assume $f_{\dot{x},\dot{y}} \leq \max \dot{y}$ and $\hat{f}_{\dot{y},\dot{x}} \leq \max \dot{x}$. We can write

$$\mathrm{wd}_f(\dot{x}, \dot{y}) = a^{f_{\dot{x},\dot{y}}} w_1 \sigma_1^{r_1} w_2 \sigma_2^{r_2} w_3 b^{\hat{f}_{\dot{y},\dot{x}}},$$

with $\{\sigma_1^{r_1}, \sigma_2^{r_2}\} = \{a^{\max \dot{y}}, b^{\max \dot{x}}\}$. Then, as $a^{f_{\dot{x},\dot{y}}} w_1 \sigma_1^{r_1} w_2 \sigma_2^{r_2}$ is a proper prefix of $\mathrm{wd}_f(\dot{x}, \dot{y})$, Lemma 1 implies $\hat{f}(\max \dot{y}) > \max \dot{x}$, which in turn implies $\max \dot{y} \geq f(\max \dot{x})$. Analogously, as $\sigma_1^{r_1} w_2 \sigma_2^{r_2} w_3 b^{\hat{f}_{\dot{y},\dot{x}}}$ is a proper suffix of $\mathrm{wd}_f(\dot{x}, \dot{y})$, we have $f(\max \dot{x}) > \max \dot{y}$, which is a contradiction.

The second claim is

$$(F2): \quad (f_{\dot{x},\dot{y}} \geq \max \dot{y} \;\; \text{implies} \;\; f_{\dot{x},\dot{y}} = f(\max \dot{x})) \;\; \text{AND} \;\; (\hat{f}_{\dot{y},\dot{x}} \geq \max \dot{x} \;\; \text{implies} \;\; \hat{f}_{\dot{y},\dot{x}} = \hat{f}(\max \dot{y})).$$

To prove this, first assume $f_{\dot{x},\dot{y}} \geq \max \dot{y}$. Then, $a^{f_{\dot{x},\dot{y}}}$ is a longest run of a's in the word $\mathrm{wd}_f(\dot{x}, \dot{y})$, and this word has a proper prefix of the form $a^{f_{\dot{x},\dot{y}}} w b^{\max \dot{x}}$. By Lemma 1, $\hat{f}(f_{\dot{x},\dot{y}}) > \max \dot{x}$ and, therefore, $f_{\dot{x},\dot{y}} \geq f(\max \dot{x})$. Also, by the definition of $f_{\dot{x},\dot{y}}$, we have $f_{\dot{x},\dot{y}} \leq f(\max \dot{x})$. Hence, $f_{\dot{x},\dot{y}} = f(\max \dot{x})$, as required. The second part of (F2) follows by symmetry.

The third claim is

$$(F3): \quad (f_{\dot{x},\dot{y}} > \max \dot{y} \;\; \text{implies} \;\; \hat{f}_{\dot{y},\dot{x}} \leq \max \dot{x}) \;\; \text{AND} \;\; (\hat{f}_{\dot{y},\dot{x}} > \max \dot{x} \;\; \text{implies} \;\; f_{\dot{x},\dot{y}} \leq \max \dot{y})$$

Due to the symmetry in (F3), we only show the second part. So assume $\hat{f}_{\dot{y},\dot{x}} > \max \dot{x}$. By (F2), $\hat{f}_{\dot{y},\dot{x}} = \hat{f}(\max \dot{y})$; hence $\hat{f}(\max \dot{y}) > \max \dot{x}$. By the properties of near inverses, $\max \dot{y} \geq f(\max \dot{x})$. Finally, by the definition of $f_{\dot{x},\dot{y}}$, we have $f_{\dot{x},\dot{y}} \leq f(\max \dot{x})$ and, therefore, $f_{\dot{x},\dot{y}} \leq \max \dot{y}$, as required.

The statement of the proposition now follows easily using the above three claims. ■

4 Finite solid codes based on f-words

For every nonempty word $v \in \{a, b\}^*$, let $\dot{x}_v$ and $\dot{y}_v$ be the unique tuples such that $\mathrm{wd}(\dot{x}_v, \dot{y}_v) = bva$. Obviously,

$$|\mathrm{wd}(\dot{x}_v, \dot{y}_v)| = |v| + 2 = \sum \dot{x}_v + \sum \dot{y}_v.$$

In [6], the authors showed that the language

$$K_n = \{a^{\max \dot{x}_v} \mathrm{wd}(\dot{x}_v, \dot{y}_v) b^{1+\max \dot{y}_v} \mid v \in \{a, b\}^n\}$$

is a solid code. Moreover, the information rate $\mathrm{rt}(K_n)$ of K_n tends to 1, as $n \to \infty$ – recall that the *information rate* $\mathrm{rt}(C)$ of a finite code C is the quantity $\log |C| / \bar{\ell}(C)$, where $\bar{\ell}(C)$ is the average word length of C. Obviously, this code contains exactly 2^n codewords. Moreover, it is evident that every binary word v of length n can be encoded to a unique codeword in linear time – the time to identify the largest runs of b's and a's in bva. Here we use the same idea and the identity function $\delta(i) = i$, whose near-inverse is $\hat{\delta}(j) = 1 + j$, to define a new sequence of solid codes that has a better information rate without sacrificing the order of magnitude of the encoding complexity.

Theorem 2 *For all $n \in \mathbb{N}$, the language*

$$F_n = \{a^{\delta_{\dot{x}_v,\dot{y}_v}} \mathrm{wd}(\dot{x}_v,\dot{y}_v)\, b^{\hat{\delta}_{\dot{y}_v,\dot{x}_v}} \mid v \in \{a,b\}^n\}$$

is a solid code of cardinality 2^n such that $\mathrm{rt}(F_n) \geq \mathrm{rt}(K_n)$ and, therefore, $\mathrm{rt}(F_n) \to 1$, as $n \to \infty$. Moreover, (F_n) has linear time encoding complexity (hence, there is an algorithm that maps every binary word of length n to a unique codeword of F_n in linear time).

Proof. First we show that F_n is a solid code. As $F_n \subseteq L_\delta$, the language F_n is overlap-free. Now suppose that F_n is not an infix code. Then there are two words

$$w = a^{\delta_{\dot{x}_v,\dot{y}_v}} \mathrm{wd}(\dot{x}_v,\dot{y}_v)\, b^{\hat{\delta}_{\dot{y}_v,\dot{x}_v}}, \;\; w' = a^{\delta_{\dot{x},\dot{y}}} \mathrm{wd}(\dot{x},\dot{y})\, b^{\hat{\delta}_{\dot{y},\dot{x}}} \;\in F_n$$

such that $w' = w_1 w w_2$, for some words w_1 and w_2 with $w_1 w_2 \neq \lambda$, and $|\mathrm{wd}(\dot{x}_v,\dot{y}_v)| = |\mathrm{wd}(\dot{x},\dot{y})| = n+2$ – of course, $\dot{x} = \dot{x}_u$ and $\dot{y} = \dot{y}_u$, for some $u \in \{a,b\}^n$, but this fact is of no use in the rest of the proof. As both of $\mathrm{wd}(\dot{x}_v,\dot{y}_v)$ and $\mathrm{wd}(\dot{x},\dot{y})$ begin with b, and w is an infix of w', we have that $\mathrm{wd}(\dot{x}_v,\dot{y}_v) = \mathrm{wd}(\dot{x},\dot{y})$, which implies that $w = w'$; a contradiction. Hence, F_n is an infix code.

The statement about the information rate of F_n follows easily from the following sequence of facts

$$\begin{aligned}
\bar{\ell}(F_n) &= \left(\sum_{u\in F_n} |u|\right)/|F_n| = \left(\sum_{v\in\{a,b\}^n} \delta_{\dot{x}_v,\dot{y}_v} + (n+2) + \hat{\delta}_{\dot{y}_v,\dot{x}_v}\right)/|F_n|, \\
\bar{\ell}(K_n) &= \left(\sum_{u\in K_n} |u|\right)/|K_n| = \left(\sum_{v\in\{a,b\}^n} \max \dot{x}_v + (n+2) + \max \dot{y}_v + 1\right)/|K_n|,
\end{aligned}$$

$|F_n| = |K_n| = 2^n$, $\delta_{\dot{x}_v,\dot{y}_v} \leq \max \dot{x}_v$ and $\hat{\delta}_{\dot{y}_v,\dot{x}_v} \leq 1 + \max \dot{y}_v$.

Regarding the encoding complexity, it is sufficient to show that $\delta_{\dot{x},\dot{y}}$ can be computed from $\mathrm{wd}(\dot{x},\dot{y})$ in linear time. Recall, $\delta_{\dot{x},\dot{y}}$ is the largest value $\delta(x_r) = x_r$ over all r with $r = 0$ or $\max(y_{k-1},\ldots,y_{k-r}) < \delta(x_r)$. The algorithm reads the runs of $\mathrm{wd}(\dot{x},\dot{y})$ left to right and uses the variables X and Y. The variable Y keeps track of the current largest length of the a-runs seen so far, and is initially zero. The variable X keeps track of the largest length of a b-run that is also greater than Y. Once the first run b^{x_0} has been read, X is set to x_0 and, then, the following loop is performed

while (there are two other runs a^i, b^j) {
  if ($Y < i$) then $Y = i$;
  if ($X < j$ AND $Y < j$) then $X = j$;
}

The final value of the variable X is equal to $\delta_{\dot{x},\dot{y}}$. ■

The improvement $\mathrm{rt}(F_n) \geq \mathrm{rt}(K_n)$ in the above result could be nontrivial. In particular there are many pairs of tuples $(\dot{x},\dot{y})$ for which the corresponding words in F_n are strictly shorter than those in K_n – again of course $\dot{x} = \dot{x}_v$ and $\dot{y} = \dot{y}_v$, for some $v \in \{a,b\}^n$. To see this consider all tuples $\dot{x},\dot{y}$ of some fixed length k such that

$$\sum \dot{x} + \sum \dot{y} = n+2, \;\; y_0 = 1, \;\; \max \dot{x} = x_{k-1} > \max \dot{y}.$$

Then $\delta_{\dot{x},\dot{y}} = x_{k-1}$ and $\hat{\delta}_{\dot{y},\dot{x}} = \hat{\delta}(y_0) = 2$ – the latter equation follows from the facts $\hat{\delta}(y_i) \leq x_{k-1}$, for all indices i, and $I_{\hat{\delta}}(\dot{y},\dot{x}) = \{0\}$. Moreover, for all of these tuples we have

$$a^{x_{k-1}} \mathrm{wd}(\dot{x},\dot{y}) b^{1+\max \dot{y}} \in K_n \;\text{ and }\; a^{x_{k-1}} \mathrm{wd}(\dot{x},\dot{y}) b^2 \in F_n.$$

Clearly, each of these F_n-words is $(\max \dot{y} - 1)$ bits shorter than the corresponding K_n-word.

Example 3 *For $n = 16$ and $k = 3$, if we consider all tuples $\dot{x}, \dot{y}$ as above such that $\max \dot{x} = x_{k-1} = 5$, $y_0 = 1$, and $\max \dot{y} = 4$ we get the equation*

$$x_0 + x_1 + y_1 + y_2 = 12, \quad \textit{with} \quad \max(x_0, x_1) \leq 5, \ \max(y_1, y_2) = 4.$$

This equation has 29 solutions, which implies that there are at least 29 words in F_{16} with each one being 3 bits shorter than the corresponding word in K_{16}.

We turn now to finite solid codes with error-detecting capabilities. For a finite nonempty language L, we use $\ell(L)$ to denote the length of the longest word in L. If all the words of L are of the same length, say n, then L is called a *uniform*, or *block*, code of length n. We recall that a *(combinatorial) channel* γ is a set of pairs of words that is domain preserving: $\gamma \subseteq \Sigma^* \times \Sigma^*$ and, for all pairs (u, v), if $(u, v) \in \gamma$ then $(u, u) \in \gamma$. When $(u, v) \in \gamma$ it is meant that, on input u, we can receive v via the channel γ. If $v \neq u$, then u is received with errors. The domain preserving requirement ensures that error-free communication via γ is always possible. The channel $\sigma(1, n)$ permits at most one substitution error in any infix of length n of the input message. For example, $(00000, 10001) \in \sigma(1, 4)$ but $(00000, 01001) \notin \sigma(1, 4)$.

Note: The expression "one error" means literally an error that involves exactly one symbol and not one word – this is a standard convention in the theory of error control codes.

A language L is error-detecting for a channel γ, if no word of $L \cup \{\lambda\}$ (the input message) can result in *another* word of $L \cup \{\lambda\}$ using the errors permitted by the channel γ: if $w, z \in L \cup \{\lambda\}$ and $(w, z) \in \gamma$, then $w = z$ – see [4], [7] for details on these concepts.

In [6], it is shown that if the code K_n is restricted as follows

$$K_C = \{a^{\max \dot{x}_v} \mathrm{wd}(\dot{x}_v, \dot{y}_v) b^{1+\max \dot{y}_v} \mid v \in C\},$$

where C is any uniform code of length n that is error-detecting for the channel $\sigma(1, n)$, then K_C^* is error-detecting for the channel that permits at most one substitution, insertion, or deletion error in any segment of length $\ell(K_C)$ of the transmitted message. This result is shown in Theorem 7.2 of [6]. Using similar ideas as in the proof of that theorem, it turns out that, for

$$F_C = \{a^{\delta_{\dot{x}_v, \dot{y}_v}} \mathrm{wd}(\dot{x}_v, \dot{y}_v) b^{\hat{\delta}_{\dot{y}_v, \dot{x}_v}} \mid v \in C\},$$

the language F_C^* is error-detecting as well.

Theorem 3 *Let C be a uniform code of length n which is error-detecting for the channel $\sigma(1, n)$. Then, the language F_C^* is error-detecting for the channel that permits at most one substitution, insertion, or deletion error in any segment of length $\ell(F_C)$ of the input message.*

Proof. Let γ_m be the channel that permits at most one substitution, insertion, or deletion error in any segment of length m of the input message, where $m \in \mathbb{N}$. The proof is based on the following two facts from [6] and [7], respectively.

(F1) For any two mappings $h_a, h_b : \{a, b\}^+ \to \mathbb{N}$, the language

$$L_C = \{a^{h_a(bva)} bva b^{h_b(bva)} \mid v \in C\}$$

is error-detecting for the channel $\gamma_{\ell(L_C)}$.

(F2) If H is a finite solid code then H^* is error-detecting for $\gamma_{\ell(H)}$ if and only if the following holds: $w \in H^*$, $z \in H \cup \{\lambda\}$ and $(w, z) \in \gamma_{\ell(H)}$ implies $w = z$.

We show that F_C^* is error-detecting for $\gamma_{\ell(F_C)}$ using fact (F2). So let $w \in F_C^*$ and $z \in F_C \cup \{\lambda\}$ such that $(w, z) \in \gamma_{\ell(F_C)}$. As the codewords of any solid code have length at least 2 and the channel cannot delete, or insert, more than one symbol in any codeword, it follows that $w = \lambda$ if and only if $z = \lambda$. Hence, $z = w$, as required, when at least one of w and z is empty. So in the rest of the proof we assume that $z \in F_C$ and $w \in F_C^m$ for some $m \in \mathbb{N}$.

If $m = 1$ again we have $w = z$, as required, this time using fact (F1). Finally we will arrive at a contradiction assuming that $m \geq 2$. First note that for each $v \in C$, we have that $|\mathrm{wd}(\dot{x}_v, \dot{y}_v)| = n+2$ and $1 \leq \delta_{\dot{x}_v,\dot{y}_v} \leq \max \dot{x}_v$ and $2 \leq \hat{\delta}_{\dot{y}_v,\dot{x}_v} \leq 1 + \max \dot{y}_v$. Moreover, $\max \dot{x}_v + \max \dot{y}_v \leq \sum \dot{x}_v + \sum \dot{y}_v = n + 2$. This implies that, for each codeword $u \in F_C$, we have

$$n + 5 \leq |u| \leq 2n + 5.$$

Now, as z is received from w and there can be at most m deletions in w, we have that $|z| \geq |w| - m$. Moreover we have that $|w| \geq m(n+5) = mn+5m$. Hence, $|z| \geq mn+4m \geq 2n+8$; a contradiction. ■

5 Infinite maximal solid codes based on f-words

In [3] the authors obtained a complete structural characterization of *all* infinite maximal solid codes that are subsets of $a^+b^+a^+b^+$. In particular, for any non-decreasing and unbounded function $f : \mathbb{N} \to \mathbb{N}$, the language

$$C_{f,1} = \{a^{f(x)}b^x a^y b^{\hat{f}(y)} \mid x, y \in \mathbb{N}\}$$

is a maximal solid code and, conversely, every maximal solid code that is a subset of $a^+b^+a^+b^+$ must be equal to $C_{f,1}$, for some (non-decreasing and unbounded) function f. It should be clear that $C_{f,1} \subseteq L_f$. Then it is natural to consider, for each $k \geq 1$, the language

$$C_{f,k} = \{a^{f_{\dot{x},\dot{y}}}b^{x_0}(a^{y_{k-1}}b^{x_1}) \cdots (a^{y_1}b^{x_{k-1}})a^{y_0}b^{\hat{f}_{\dot{y},\dot{x}}} \mid \dot{x}, \dot{y} \in \mathbb{N}^k\}.$$

As this language is a subset of L_f, it is an overlap-free language. Moreover, it is not difficult to see that $C_{f,k}$ is an infix code and, therefore, a solid code. It turns out, however, that $C_{f,k}$ is not a maximal solid code, and it appears that a complete characterization of maximal solid codes containing $C_{f,k}$ is rather difficult to describe. In the sequel we show that, for any choice of the (non-decreasing and unbounded) function f, the language

$$K_{f,2} = C_{f,2} \cup \{\mathrm{wd}_f(\dot{s}, \dot{t}) \mid \dot{s}, \dot{t} \in \mathbb{N}^3,\ f(s_1) > t_2,\ \hat{f}(t_1) > s_2\}$$

is a maximal solid code. The words of this language are of the form

$$a^{f_{\dot{s},\dot{t}}}b^{s_0}a^{t_1}b^{s_1}a^{t_0}b^{\hat{f}_{\dot{t},\dot{s}}} \quad \text{or} \quad a^{f_{\dot{s},\dot{t}}}b^{s_0}a^{t_2}b^{s_1}a^{t_1}b^{s_2}a^{t_0}b^{\hat{f}_{\dot{t},\dot{s}}}. \tag{13}$$

In the sequel, we need to work with words w of the form

$$w = a^{y_m}\, b^{x_0}a^{y_{m-1}}b^{x_1} \cdots a^{y_{m-i}}b^{x_i} \cdots a^{y_1}b^{x_{m-1}}a^{y_0}\, b^{x_m}, \tag{14}$$

where $\dot{x}$ and $\dot{y}$ are tuples of length $m+1$. Obviously, any f-word is of the form shown in (14). Now we define the *pattern* (with respect to f)

$$\tilde{w} = P_0P_1 \cdots P_m$$

of w as follows. Each P_i is either the symbol F or G, depending on whether $\hat{f}(y_{m-i}) > x_i$ or $y_{m-i} < f(x_i)$, respectively. Equivalently, P_i is either the symbol F or G, depending on whether $y_{m-i} \geq f(x_i)$ or $\hat{f}(y_{m-i}) \leq x_i$, respectively. For example, for $f(i) = i$ and $\hat{f}(j) = 1 + j$, we have that the pattern of $a^3b^2a^2b^3ab^2$ is FGG. By the definition of $f_{\dot{s},\dot{t}}$ and $\hat{f}_{\dot{t},\dot{s}}$, it is easy to see that the pattern of any f-word starts with F and ends with G.

When P_i = F and P_{i+1} = G, we write $(P_i > P_{i+1})$ if $y_{m-i} \geq f(x_{i+1})$, and $(P_i < P_{i+1})$ if $\hat{f}(y_{m-i}) \leq x_{i+1}$. Obviously, by the properties of near-inverses, exactly one of $(P_i > P_{i+1})$ and $(P_i < P_{i+1})$ is true. The *enhanced pattern* $\vec{w}$ of w is the expression that results when we replace in $\tilde{w}$ each FG with (F>G) or (F<G), depending on which of the two expressions is true. For example, the enhanced pattern of the word $a^3b^2a^2b^3ab^2$ is (F>G)G.

Lemma 2 *Let w be a word of the form shown in (14).*

1. *If $\tilde{w}$ begins with G then w begins with a nonempty suffix of a word in $K_{f,2}$.*
2. *If $\tilde{w}$ ends with F then w ends with a nonempty prefix of a word in $K_{f,2}$.*
3. *If $\vec{w}$ begins with (F<G) then w begins with a nonempty suffix of a word in $K_{f,2}$.*
4. *If $\vec{w}$ ends with (F>G) then w ends with a nonempty prefix of a word in $K_{f,2}$.*
5. *If F(F<G), or (F>G)G, or (F>G)(F<G) is an infix of $\vec{w}$ then a word of $K_{f,2}$ is an infix of w.*

Proof. The proof of each statement is based on the definitions of pattern and enhanced pattern, using the forms of w and the words in $K_{f,2}$ as shown in (14) and (13), respectively. Moreover, we use the facts that $1 \in I_f(\dot{s},\dot{t})$ implies $f_{\dot{s},\dot{t}} \geq f(s_1)$, and $1 \in I_{\hat{f}}(\dot{t},\dot{s})$ implies $\hat{f}_{\dot{t},\dot{s}} \geq \hat{f}(t_1)$.

For the first statement, assume that w begins with $a^{y_m}b^{x_0}$ such that $\hat{f}(y_m) \leq x_0$. The statement follows when we note that the prefix $a^{y_m}b^{\hat{f}(y_m)}$ of w is a suffix of $a^{f(1)}b^1a^1b^1a^{y_m}b^{\hat{f}(y_m)} \in K_{f,2}$. The second statement is symmetric to the first one.

For the third statement, assume that w begins with the word $a^{y_m}b^{x_0}a^{y_{m-1}}b^{x_1}$ whose enhanced pattern is F<G. Then, $\hat{f}(y_m) > x_0$, $\hat{f}(y_{m-1}) \leq x_1$, and $\hat{f}(y_m) \leq x_1$. This implies that the word

$$v = a^{f(1)}b^1a^{y_m}b^{x_0}a^{y_{m-1}}b^{\max(\hat{f}(y_m),\hat{f}(y_{m-1}))}$$

is in $K_{f,2}$ and that w begins with nonempty suffix of v, as required. The fourth statement is symmetric to the third one.

For the fifth statement, we only show the case (F > G)(F < G) – the other two cases can be handled using similar arguments. So assume that w has an infix of the form

$$a^{t_3}b^{s_0}a^{t_2}b^{s_1}a^{t_1}b^{s_2}a^{t_0}b^{s_3}$$

whose enhanced pattern is (F>G)(F<G). Then, $t_3 \geq f(s_0)$, $t_3 \geq f(s_1)$, $t_2 < f(s_1)$, and $\hat{f}(t_1) > s_2$, $\hat{f}(t_1) \leq s_3$, $\hat{f}(t_0) \leq s_3$. Hence, the word

$$a^{\max(f(s_0),f(s_1))}b^{s_0}a^{t_2}b^{s_1}a^{t_1}b^{s_2}a^{t_0}b^{\max(\hat{f}(t_0),\hat{f}(t_1))}$$

is in $K_{f,2}$ and an infix of w, as required. ■

Theorem 4 *For every non-decreasing and unbounded function $f : \mathbb{N} \to \mathbb{N}$, the language $K_{f,2}$ is a maximal solid code.*

Proof. First we show that $K_{f,2}$ is indeed a solid code. As it is a subset of L_f, $K_{f,2}$ is an overlap-free language. To see that it is also an infix code we note that the pattern of any word in $(K_{f,2} - a^+b^+a^+b^+a^+b^+)$ is equal to FGFG, and the pattern of any word in $K_{f,2} \cap a^+b^+a^+b^+a^+b^+$ is FGG or FFG.

Next we show the maximality of $K_{f,2}$ using contradiction. So assume there is a word $w \notin K_{f,2}$ of the form shown in (14) such that $K_{f,2} \cup \{w\}$ is a solid code. First we note that, by Lemma 2(1-2), the pattern $\tilde{w}$ must begin with F and end with G. Now we claim that $\tilde{w}$ can contain neither FF nor GG. With this claim $\tilde{w}$ must be of the form $\mathrm{FG}\cdots\mathrm{FG}$, and then by Lemma 2(3-4), $\vec{w}$ must begin with (F>G) and end with (F<G). This leads to the desired contradiction via Lemma 2(5). Now we prove the FF case of our claim in the following paragraph (the other case is symmetric).

Assume that $\tilde{w}$ contains FF. As it ends with G, it must also contain FFG. We consider the last occurrence of FFG in $\tilde{w}$. By Lemma 2(5), the enhanced pattern of w is of the form

$$\vec{w} = \vec{w}_1\mathrm{F}(\mathrm{F} > \mathrm{G})\vec{w}_2,$$

with no occurrence of FFG in $\tilde{w}_2$. Now let r be the largest nonnegative integer such that $\vec{w}_2$ is of the form $(\mathrm{F} > \mathrm{G})^r\vec{w}_3$; hence, $\vec{w}_3$ cannot begin with (F > G). Moreover, by Lemma 2(5), $\vec{w}_3$ cannot begin with (F < G). Hence, $\tilde{w}_3$ cannot begin with FG. By Lemma 2(4), $\vec{w}_3$ must be nonempty. So $\vec{w}_3$ must begin with F, in view of Lemma 2(5). Moreover, as $\tilde{w}_3$ ends with G and contains no FFG, it must begin with FG, which is impossible. ■

6 Discussion

We have presented new classes of both, finite and infinite, binary solid codes based on word runs. These utilize the new concept of f-word. Our work constitutes another step towards the explicit structural characterization of large classes of binary solid codes satisfying various good criteria. Can we find a reasonable characterization of all maximal binary solid codes that are subsets of $(a^+b^+)^k$, for any $k \geq 3$? Then, of such codes that are subsets of $(a^+b^+)^+$? Would such constructions also lead to improved finite solid codes?

Acknowledgement

The authors are thankful to the anonymous referees for providing detailed and constructive comments on the submitted version of this paper.

References

[1] V.B. Balakirsky. Block codes for asynchronous data transmission designed from binary trees. *The Computer Journal* **45**:2 (2002), 243–248.

[2] H. Jürgensen. Markers and deterministic acceptors for non-deterministic languages. *J. Automata, Languages and Combinatorics* **14**:1 (2009), 33–62.

[3] H. Jürgensen, M. Katsura and S. Konstantinidis. Maximal solid codes. *J. Automata, Languages and Combinatorics* **6** (2001), 25–50.

[4] H. Jürgensen and S. Konstantinidis. Codes. In [11], Vol. 1, pp 511–607.

[5] H. Jürgensen and S. Konstantinidis. (Near-)inverses of sequences. *International J. of Computer Mathematics* **83**:2 (2006), 203–222.

[6] H. Jürgensen, S. Konstantinidis and N.H. Lâm. Asymptotically optimal low-cost solid codes. *Journal of Automata, Languages and Combinatorics* **9**:1 (2004), 81–102.

[7] S. Konstantinidis and A. O'Hearn. Error-detecting properties of languages. *Theoretical Computer Science* **276**:1-2 (2002), 355–375.

[8] N.H. Lâm. Finite maximal solid codes. *Theoretical Computer Science* **262**:1-2 (2001), 333–347.

[9] V.I. Levenshtein. Maximum number of words in codes without overlaps. *Probl. Inform. Transm.* **6** (1970), 355–357.

[10] V.I. Levenshtein. Combinatorial problems motivated by comma-free codes. *J. Combinatorial Designs* **12** (2004), 184–196.

[11] G. Rozenberg and A. Salomaa (eds). *Handbook of Formal Languages, Vol. 1.* Springer-Verlag, Berlin, 1997.

[12] Y-S. Han and D. Wood. Overlap-free regular languages. *In: COCOON 2006, Lecture Notes in Computer Science* **4112** (2006), 469–478.

[13] S.S. Yu. *Languages and Codes.* Tsang Hai Publishing Co., Taichung, Taiwan, 2005.

(Received: October 18, 2009; revised: April 27, 2010)