INFORMATION AND COMPUTATION 96, 55–64 (1992)

Retrograde Codes and Bounded Synchronization Delay

HOLLIE L. BUCHANAN AND MICHAEL E. MAYS

Department of Mathematics,
West Virginia University, Morgantown, West Virginia 26506

We study how the concept of bounded synchronization delay is related to retrograde comma-free codes. Retrograde codes are a subclass of comma-free codes in which the dictionary of code words excludes not only overlaps of code words but also reversals of overlaps. We give a general upper bound on the maximum size of a retrograde comma-free distionary, provide a construction for a bounded synchronization delay retrograde code which attains this maximum size, and discuss traditional dictionaries in the context of retrograde codes.

I. INTRODUCTION

Comma-free codes were first introduced in (Crick, Griffith, and Orgel, 1957) in connection with a conjectured structure for DNA. For both comma-free codes and the more general bounded synchronization delay codes, it is desired to restrict a potential dictionary of codewords so that "framing errors" are avoided. In the case of comma-free codes, this means that overlaps of codewords are excluded from the dictionary. The mathematical development of the theory in (Golomb, Welch, and Delbrück, 1958) was guided by the potential benefit of enumeration and classification of comma-free codes to understanding the structure of DNA. Although it became apparent that the structure provided by comma-free codes was not the primary mechanism for avoiding framing errors in DNA, the concept was still of independent interest. General topics considered included the maximum size of a comma-free dictionary as a function of word length k and alphabet size n (Golomb, Gordon, and Welch, 1958, Jiggs, 1963), construction techniques for building dictionaries of maximum size ((Eastman, 1965) and (Scholtz, 1966)), and variations involving less restrictive conditions such as dictionaries allowing words of variable length (Scholtz, 1969) and codes with bounded synchronization delay (Golomb and Gordon, 1965). An advantage to working with codes of bounded synchronization delay is that the upper bound on maximum dictionary size obtained by a simple combinatorial argument is always achieved. A nice survey of results about synchronizable codes is contained in (Cummings, 1987).

One special class of comma-free codes that was considered in (Golomb,

0890-5401/92 $3.00

Welch, and Delbrück, 1958) was the class of transposable codes. Just as in a comma-free code the dictionary of codewords is chosen to rule out overlaps, for transposable codes not only are overlaps ruled out but also reversals of "complementary words."

More precisely, suppose a dictionary of k letter words is being constructed from an n letter alphabet $\{0, 1, 2, ..., n-1\}$. The dictionary is comma-free if, for $a_1 a_2 \cdots a_k$ and $b_1 b_2 \cdots b_k$ any two words in the dictionary, the words $a_2 a_3 \cdots a_k b_1$, $a_3 \cdots a_k b_1 b_2$, ..., $a_k b_1 \cdots b_{k-1}$ are not in the dictionary. Now assume that n is even, and consider a fixed permutation σ of the alphabet with the property that σ can be written as a product of $n/2$ disjoint transpositions. A comma-free dictionary is transposable with respect to σ if, for $a_1 a_2 \cdots a_k$ and $b_1 b_2 \cdots b_k$ any two words in the dictionary, the words $\sigma(a_k) \cdots \sigma(a_2)\,\sigma(a_1)$, $\sigma(b_1)\,\sigma(a_k) \cdots \sigma(a_2)$, $\sigma(b_2)\,\sigma(b_1)\,\sigma(a_k) \cdots \sigma(a_3)$, ..., $\sigma(b_{k-1}) \cdots \sigma(b_1)\,\sigma(a_k)$ are excluded from the dictionary.

A related idea was introduced in (Mays, 1987). A retrograde code excludes overlaps and reversals of overlaps: $a_1 a_2 \cdots a_k$ and $b_1 b_2 \cdots b_k$ in the dictionary preclude the appearance of any of $a_k a_{k-1} \cdots a_2 a_1$, $b_1 a_k \cdots a_2$, $b_2 b_1 \cdots a_3$, ..., $b_{k-1} \cdots b_1 a_k$. Thus the permutation σ in the transposable case is taken in this case to be the identity. A comma-free retrograde code built with k letter words is able to distinguish both direction and breakpoints of a message after at most $2k-1$ symbols have been transmitted. If a word and its reversal were both allowed in the dictionary, the "weakly retrograde" code resulting would in the worst case be able to distinguish breakpoints but not direction.

Comma-free codes require only $2k-2$ symbols to determine breakpoints, since if a code word has not appeared within $2k-2$ symbols the breakpoint must be in the middle. To see that the weaker bound of $2k-1$ symbols is correct for retrograde codes, consider the dictionary of size 10, consisting of $k=3$ letter words built from an alphabet of $n=5$ symbols

$$\{201, 301, 401, 302, 402, 312, 412, 403, 413, 423\}$$

and the message ... 0230 .... It is clear that the breakpoint must be between 2 and 3, but we need to see one more symbol before the direction is determined.

II. A Bound on the Size of a Retrograde Dictionary

We write $W_k(n)$ as the size of a maximal comma-free dictionary built of k letter words from an alphabet with n symbols and $R_k(n)$ as the size of a maximal retrograde comma-free dictionary. The standard bound on

$W_k(n)$ is obtained by applying the Möbius inversion formula to a sorting of the n^k possible words into equivalence classes of cyclic permutations of letters:

$$W_k(n) \leqslant \frac{1}{k} \sum_{d \mid k} \mu\left(\frac{k}{d}\right) n^d.$$

Here the Möbius function μ arises in its usual inclusion–exclusion context, and the summation is over divisors d of k.

The argument works for synchronizable codes as well as comma-free codes: the bound is achieved for arbitrary odd k in the comma-free case, and achieved for arbitrary k for the synchronizable case. In (Mays, 1987) it was shown that, if k is odd,

$$R_k(n) \leqslant \frac{1}{2k} \sum_{d \mid k} \mu\left(\frac{k}{d}\right) (n^d - kn^{(d+1)/2}).$$

The goal now is to develop a more general result to include the case when k is even.

We consider three types of words that can never occur in any retrograde synchronizable dictionary. First, in analogy with the comma-free codes, we say a word of length k has period d, where $d \mid k$, if the word consists of a subword d symbols long repeated k/d times. A typical word of period d could be written $a_1 a_2 \cdots a_d a_1 a_2 \cdots a_d \cdots a_1 a_2 \cdots a_d$. Then words of period d, where d is a proper divisor of k, are ruled out because a single word of period d endlessly repeated has ambiguous breakpoints. Second, palindromes, words that read the same forwards and backwards, cannot occur, nor can cyclic permutations of palindromes (CPPs). Palindromes have an obvious ambiguity in direction, and a repeated CPP can be read backwards with shifted breakpoints. Last, if k is even, words that are concatenations of two odd palindromes (CTOPs) cannot occur for the same reason that CPPs cannot occur. Note that a cyclic permutation of a CTOP is again a CTOP, and that there is always a cyclic permutation of a CTOP that results in a single letter followed by a palindrome of length $k-1$. We use these $1, k-1$ CTOPs as canonical representatives of equivalence classes of CTOPs.

We will build a bound by arguing that once words in these three classes are ruled out, the remaining words, which are called "primitive," are partitioned into equivalence classes of size $2k$ by counting a word as being equivalent to a cyclic permutation of its letters or a reversal of a cyclic permutation of its letters. Then the largest possible dictionary would consist of one representative from each equivalence class of primitive words.

Denote by $A_d(n)$ the number of palindromes of period d, by $B_d(n)$ the number of $1, k-1$ CTOPs of period d, and by $P_d(n)$ the number of words of period d which are neither CPPs nor CTOPs. The number of primitive words is then given by $P_k(n)$.

LEMMA 1. $\sum_{\delta \mid d} A_\delta(n) = n^{\lfloor (1+d)/2 \rfloor}$.

Proof. A palindrome of period dividing d is determined by its first $\lfloor (1+d)/2 \rfloor$ positions, each of which can be filled in n ways.

LEMMA 2. *A* $1, k-1$ *CTOP of period d has the form* $aP_{d-1}aP_{d-1} \cdots aP_{d-1}$, *where* P_{d-1} *is a palindrome of length* $d-1$.

Proof. A word of period d repeats the first d symbols, say $ab_1b_2 \cdots b_{d-1}$. These are the last d symbols of the word as well. But the word being a $1, k-1$ CTOP means that the last $k-1$ symbols are $b_{d-1} \cdots b_2b_1$. Hence the string $b_1b_2 \cdots b_{k-1}$ is a palindrome.

LEMMA 3.

$$\sum_{\delta \mid d} B_\delta(n) = \begin{cases} n^{1+\lfloor d/2 \rfloor}, & k \text{ even} \\ 0, & k \text{ odd}. \end{cases}$$

Proof. There are no CTOPs if k is odd. If k is even, by Lemma 2 a $1, k-1$ CTOP of period dividing d is determined by its first $1 + \lfloor d/2 \rfloor$ positions, each of which can be filled in n ways.

LEMMA 4. *The only words that are both palindromes and* $1, k-1$ *CTOPs are the n constant words.*

Proof. Suppose a word $a_1a_2 \cdots a_k$ is both a palindrome and a $1, k-1$ CTOP. $a_1 = a_k$ since the word is a palindrome, $a_k = a_2$ since the word is a $1, k-1$ CTOP, $a_2 = a_{k-1}$ since the word is a palindrome, and so on, stepping back and forth towards the middle of the word from both ends.

LEMMA 5.

$$A_k(n) = \sum_{d \mid k} \mu\left(\frac{k}{d}\right) n^{\lfloor (d+1)/2 \rfloor}.$$

Proof. Apply Möbius inversion to the formula of Lemma 1.

LEMMA 6.

$$B_k(n) = \frac{1+(-1)^k}{2} \sum_{d \mid k} \mu\left(\frac{k}{d}\right) n^{\lfloor d/2 + 1 \rfloor}.$$

Proof. Apply Möbius inversion to the formula of Lemma 3, using the $(1+(-1)^k)/2$ factor to incorporate the fact that $B_k(n)$ is 0 if k is odd.

THEOREM 1.

$$R_k(n) \leqslant \frac{1}{2k} \sum_{d \mid k} \mu\left(\frac{k}{d}\right)\left(n^d - \frac{k}{(2,k)}\left(n^{\lfloor (d+1)/2 \rfloor} + \frac{1+(-1)^k}{2} n^{\lfloor 1+d/2 \rfloor}\right)\right)$$

Proof. We partition the universe of n^k words into equivalence classes by saying that two words are equivalent if one is a cyclic permutation or a reversal of a cyclic permutation of the other, and consider the sizes and possible representatives. If d is odd, a palindrome of period d is in a class with $d-1$ other words, none of them palindromes. If d is even, after $d/2$ cyclic shifts there is another palindrome that arises (distinct from the original, else the original was of period $d/2$ to begin with). Hence these equivalence classes use up two palindromes per class. There is a similar observation to be made about $1, k-1$ CTOPs of period d. If d is odd, $d-1$ is even, and there are d cyclic shifts before we cycle back to a $1, k-1$ CTOP (the original one). If d is even, $d-1$ is odd, and after $d/2$ cyclic shifts we get a $1, k-1$ CTOP again. Thus the equivalence classes of $1, k-1$ CTOPs have two $1, k-1$ CTOPs per class if d is even.

Since every non-constant word is counted by exactly one of $A_d(n)$, $B_d(n)$, or $P_d(n)$, for some d, we have

$$n^k + n = \sum_{d \mid k} P_d(n) + \sum_{\substack{d \mid k \\ d \text{ odd}}} dA_d(n) + \sum_{\substack{d \mid k \\ d \text{ even}}} dA_d(n)/2$$

$$+ \sum_{\substack{d \mid k \\ d \text{ odd}}} dB_d(n) + \sum_{\substack{d \mid k \\ d \text{ even}}} dB_d(n)/2.$$

The extra term n on the right hand side is there because the n constant words are counted by both $A_1(n)$ and $B_1(n)$. Lemma 1 guarantees that this is the only overlap between CPPs and CTOPs. Thus

$$n^k + n = \sum_{d \mid k} \left(P_d(n) + \frac{d}{(2,d)} A_d(n) + \frac{d}{(2,d)} B_d(n)\right).$$

Apply Möbius inversion to obtain

$$P_k(n) + \frac{k}{(2,k)} (A_k(n) + B_k(n)) = \sum_{d \mid k} \mu\left(\frac{k}{d}\right)(n^d + n).$$

Since the theorem is trivially true if $k = 1$, we may take $k > 1$ to write

$$\sum_{d \mid k} \mu\left(\frac{k}{d}\right) n = n \sum_{d \mid k} \mu\left(\frac{k}{d}\right) = 0.$$

Thus

$$P_k(n) = \sum_{d \mid k} \mu\left(\frac{k}{d}\right) n^d - \frac{k}{(2, k)} (A_k(n) + B_k(n)).$$

Now substitute for $A_k(n)$ and $B_k(n)$ from Lemmas 5 and 6. The theorem follows by noting that the equivalence classes of primitive words of period k are of size $2k$, and at most one word per equivalence class may be chosen for a retrograde dictionary.

III. Bounded Delay Retrograde Codes

We introduce bounded synchronization delay (BSD) retrograde codes for the same reason that BSD codes were introduced in the study of comma-free codes: the upper bound given by the naive combinatorial argument is always achieved. A BSD retrograde code for an alphabet of size n and words of length k is a collection of words (called a BSD dictionary) with the property that it is possible to determine both the positions of word breaks and the message direction after examining some finite number s of letters. Denote the bound on $R_k(n)$ obtained in Theorem 1 by $D_k(n)$, and the size of the largest possible dictionary for a BSD retrograde code by $BSD_k(n)$.

Theorem 2. $BSD_k(n) = D_k(n)$.

Proof. That $BSD_k(n) \leqslant D_k(n)$ follows from the proof of Theorem 1. Words of period d (where d is a proper divisor of k), CPPs, and CTOPs have to be ruled out for any synchronizable retrograde code, and at most one word from each equivalence class of primitive words is allowed. To see that $BSD_k(n) \geqslant D_k(n)$, we construct a particular BSD retrograde dictionary of size $D_k(n)$ is the following manner: put the n alphabet symbols which make up the words in some convenient order, and then from each of the $D_k(n)$ equivalence classes of primitive words choose as a dictionary entry that word which is lexicographically least. Further, label the words in the dictionary to order them lexicographically as $w_1 < w_2 < \cdots < w_D$.

We now verify the bounded delay property for this dictionary by showing that neither ambiguity of breakpoint nor ambiguity of direction can persist.

For ambiguity of breakpoint, consider words $w_i = a_1 a_2 \cdots a_k$, $w_j = b_1 b_2 \cdots b_k$, and w_p contained in the overlap of w_i and w_j, say $w_p = a_{v+1} \cdots a_k b_1 \cdots b_v$. First observe that $i \neq j$, since if $i = j$ then w_p is a cyclic permutation of w_i, and only one of w_i and w_p can be in the dictionary. Next, if $i > j$, then $w_i > w_j$, and $a_1 a_2 \cdots a_v \cdots a_k > b_1 b_2 \cdots b_v \cdots b_k$ implies $a_1 a_2 \cdots a_v \geqslant b_1 b_2 \cdots b_v$. If equality holds, then w_p is a cyclic permutation of w_i again, so in fact we have $a_1 a_2 \cdots a_v > b_1 b_2 \cdots b_v$. Since the dictionary representatives were chosen to be lexicographically least, $w_i = a_1 a_2 \cdots a_k < a_{v+1} \cdots a_k a_1 \cdots a_v$, hence $a_1 a_2 \cdots a_{k-v} \leqslant a_{v+1} \cdots a_k$. Again, equality cannot occur since w_i was the smallest word in its equivalence class, so $a_1 a_2 \cdots a_{k-v} < a_{v+1} \cdots a_k$. Letting $u = \min(v, k-v)$, we obtain that $a_{v+1} \cdots a_{u+v} > b_1 b_2 \cdots b_u$, and in particular, $w_p = a_{v+1} \cdots a_k b_1 \cdots b_v > b_1 \cdots b_v a_{v+1} \cdots a_k$. Since $a_{v+1} \cdots a_k b_1 \cdots b_v$ is equivalent to $b_1 b_2 \cdots b_v a_{v+1} \cdots a_k$, we must have w_p not in the dictionary.

We have thus shown that if w_p is a dictionary entry which lies in the overlap of w_i and w_j, then $i < j$. Therefore if we have an ambiguous message M with decodings

$$\cdots w_{j_1} w_{j_2} w_{j_3} \cdots w_{j_t} \cdots$$

and

$$\cdots w_{p_1} w_{p_2} \cdots w_{p_{t-1}} \cdots$$

then

$$1 < j_1 < p_1 < j_2 < p_2 < \cdots < j_{t-1} < p_{t-1} < j_t < D,$$

since there can be no ambiguity concerning w_1 or w_D. This implies $2t - 1 \leqslant D - 2$, hence $|M| \leqslant t_k \leqslant kD/2$. We note that this argument applies in the non-retrograde case as well, and first arises in (Golomb and Gordon, 1965).

Now we consider ambiguity of direction. Suppose there exist words in the dictionary

$$\begin{aligned} w_a &= a_1 a_2 \cdots a_k, \\ w_b &= b_1 b_2 \cdots b_k, \\ w_c &= c_1 c_2 \cdots c_k, \\ w_d &= c_v c_{v-1} \cdots c_1 b_k b_{k-1} \cdots b_{v+1}, \qquad \text{and} \\ w_e &= b_v b_{v-1} \cdots b_1 a_k a_{k-1} \cdots a_{v+1}. \end{aligned}$$

This would be the situation if a segment of a message could be decoded ambiguously:

$$\begin{array}{c} \rightarrow w_a \rightarrow \qquad\qquad \rightarrow w_b \rightarrow \qquad\qquad \rightarrow w_c \rightarrow \\ a_1 a_2 \cdots a_v a_{v+1} \cdots a_k b_1 b_2 \cdots b_v b_{v+1} \cdots b_k c_1 c_2 \cdots c_v c_{v+1} \cdots c_k . \\ \leftarrow w_e \leftarrow \qquad \leftarrow w_d \leftarrow \end{array}$$

Since w_e is in the dictionary, we know $w_e = b_v b_{v-1} \cdots b_1 a_k a_{k-1} \cdots a_{v+1} < b_1 b_2 \cdots b_{v-1} b_v a_{v+1} \cdots a_k$, since the latter word is a reversal of a cyclic permutation of the former, so $b_v b_{v-1} \cdots b_1 \leqslant b_1 b_2 \cdots b_v$. Similarly, since w_b is in the dictionary, we know

$$w_b = b_1 b_2 \cdots b_v b_{v+1} \cdots b_k < b_v b_{v-1} \cdots b_1 b_k \cdots b_{v+1},$$

since the latter word is in the same equivalence class as the former. Thus $b_1 b_2 \cdots b_v \leqslant b_v b_{v-1} \cdots b_1$. Hence $b_1 b_2 \cdots b_v$ is a palindrome. The same argument yields $c_1 \cdots c_v = c_v \cdots c_1$.

Now since w_b is in the dictionary, we know

$$\begin{aligned} w_b &= b_1 b_2 \cdots b_v b_{v+1} \cdots b_k \\ &= b_v b_{v-1} \cdots b_1 b_{v+1} \cdots b_k < b_1 b_2 \cdots b_v b_k b_{k-1} \cdots b_{v+1} \end{aligned}$$

(the reversal of a cyclic permutation), so $b_{v+1} b_{v+2} \cdots b_k < b_k b_{k-1} \cdots b_{v+1}$. Similarly,

$$\begin{aligned} w_d &= c_v c_{v-1} \cdots c_1 b_k b_{k-1} \cdots b_{v+1} \\ &= c_1 c_2 \cdots c_v b_k b_{k-1} \cdots b_{v+1} \\ &< c_v c_{v-1} \cdots c_1 b_{v+1} b_{v+2} \cdots b_k \end{aligned}$$

and $b_k b_{k-1} \cdots b_{v+1} < b_{v+1} b_{v+2} \cdots b_k$, a contradiction. Thus we find that a message can have ambiguous direction only if $|M| < 3k$.

Our bound on message length for determining breakpoints exceeds our bound for determining direction if the dictionary has at least 6 words in it. Dictionary size grows rapidly with word length or alphabet size: $BSD_6(3) = 37$ already. The dictionary constructed in this case for alphabet $\{0, 1, 2\}$ consists of the words {000012, 000102, 000112, 000122, 001011, 001012, 001021, 001022, 001102, 001112, 001121, 001122, 001202, 001212, 001222, 002012, 002022, 002122, 010112, 010122, 011012, 011112, 011121, 011122, 011202, 011212, 011221, 011222, 012022, 012112, 012122, 012202, 012212, 012222, 021122, 021222, 112122}. Note that, although this dictionary satisfies the BSD property, it is not comma-free. For example, the (backwards) overlap 000012 is allowed in the sequence 001021 000012. No dictionary entry has initial letters 0100, however, so that assignment of breakpoint and direction based on that sequence of 12 symbols received would not be reasonable. The only possible assignment of breakpoint and direction in this example is the correct one.

Comparing the 37 words in this retrograde dictionary with the 116 words possible in a dictionary if direction were given, it might be a worry that the relative cost of the retrograde property would become prohibitive as word length or alphabet size grow. However, asymptotically few words are ruled out as CPPs or CTOPs, and the dominant factor in the relative cost comes from the fact that equivalence classes have $2k$ words in them, corresponding to "dihedral permutations," not the k words corresponding to cyclic permutations in case direction is unambiguous. For example, with alphabet size 6 and word length 7, the 19350 words in a maximal retrograde dictionary are close to 50% of the 39990 words possible in a dictionary for which direction is unambiguous.

IV. Adaptations of Traditional Dictionaries

One approach to building a comma-free dictionary is to devote one space per word (say the last space) to a symbol from the alphabet (call it a comma) which is used only to terminate words, and to allow any other symbol in any other position of the word. Then of the n^k words which could be built not subject to these restrictions, $(n-1)^{k-1}$ will still be allowed. It is inefficient, but acceptable, to build comma-free codes with commas. This scheme does not give enough information to distinguish both direction and breakpoints. In this section we describe three ways to build traditional dictionaries which are able to distinguish direction and breakpoints.

An $(n-2)^{k-2}$ solution: Devote two spaces per word (say the last two) and two alphabet symbols (say a and b) to marking breakpoints. Then each word of the dictionary ends in ab, and any message which is received with a ba in it has to be decoded from right to left rather than from left to right.

An $(n-1)^{k-3}$ solution: Devote three spaces per word and one alphabet symbol to marking breakpoints in the following way. If k is at least 6, let every word of the dictionary end in the pattern of symbols $\cdots axaa$, where a is allowed to occur only in these positions and x is any alphabet symbol. Then any message with the sequence $\cdots aaxa \cdots$ occurring has to be interpreted from right to left.

An $(n-3)^{k-1}$ solution: Devote three alphabet symbols (say a, b, and c) and one space per word (the last) to marking breakpoints, with the convention that any of a, b, and c can be used as the last symbol of a word, but in any message a word ending in a must be followed by a word ending in b, which must in turn be followed by a word ending in c. Then a message arriving with terminators which violate this sequence should be read backwards. This restriction amounts to imposing a "rule of grammar" on messages built from dictionary entries, and perhaps it is not in the spirit of the other two schemes proposed for this reason.

Certainly the natural combinatorial question shifts from finding the largest dictionary size to determining how many messages of a given length can be constructed. In this case a more complicated pattern could be devised to make do with just two terminators (say one word ending in *a*, then two words ending in *b*, then three ending in *a*, then one in *b*, then three ending in *a*, then one in *b*, then two in *a*, then three in *b*, and so on) at the expense of a longer orientation delay and a more complicated grammar.

RECEIVED January 25, 1988; FINAL MANUSCRIPT RECEIVED May 23, 1990

REFERENCES

CRICK, F. H. C., GRIFFITH, J. S., AND ORGEL, L. E. (1957), Codes without commas, *Proc. Nat. Acad. Sci. U.S.A.* **43**, 416.

CUMMINGS, L. J. (1987), Aspects of synchronizable coding, *J. Combin. Math. Combin. Comput.* **1**, 67.

EASTMAN, W. L. (1965), On the construction of comma-free codes, *IEEE Trans. Inform. Theory* **11**, 263.

JIGGS, B. H. (1963), Recent results in comma-free codes, *Canad. J. Math.* **15**, 178.

GOLOMB, S. W., AND GORDON, B. (1965), Codes with bounded synchronization delay, *Inform. Contr.* **8**, 355.

GOLOMB, S. W., GORDON, B., AND WELCH, L. R. (1958), Comma-free codes, *Canad. J. Math.* **10**, 202.

GOLOMB, S. W., WELCH, L. R., AND DELBRÜCK, M. (1958), Construction and properties of comma-free codes, *Biol. Medd. K. Dan. Vidensk. Selsk.* **23**, 3.

MAYS, M. E. (1987), Retrograde comma-free codes, *Congr. Numer.* **59**, 217.

SCHOLTZ, R. A. (1966), Codes with synchronization capability, *IEEE Trans. Inform. Theory* **12**, 135.

SCHOLTZ, R. A. (1969), Maximal and variable word length comma-free codes, *IEEE Trans. Inform. Theory* **15**, 300.