The Synchronization of Variable-Length Codes

Mark R. Titchener

Abstract—Many variable-length codes exhibit a tendency for resynchronization to occur automatically following any error. However, attempts to identify an underlying synchronization mechanism, and to accurately predict the expected synchronization delay (ESD), for even quite specific variable-length codes, appear to have been largely unsuccessful. The present paper explores a novel method for estimating the synchronization performance for a wide variety of variable-length codes, based on the T-Codes. T-Codes are a class of self-synchronizing codes, which typically synchronize within 2–3 codewords by a mechanism that derives from a recursive T-augmentation construction. It is observed that the T-Code mechanism for synchronization is followed, more or less, by other variable-length codes wherever substantial numbers of codewords are shared with a T-Code set. T-augmentation itself provides a means for assessing the contribution individual codewords make to the overall synchronization process for a T-Code set. Thus codeword differences between sets may be specifically evaluated to estimate the synchronization performance of a variable-length code set from a closely related T-Code set.

Index Terms—Variable-length codes, synchronization, self-synchronization, T-Codes.

I. INTRODUCTION

It has been long recognized [1], [2], that variable-length codes can exhibit pronounced self-synchronization capabilities, i.e., that the decoder will, in the course of normal decoding, achieve correct word alignment with respect to the encoding. Self-synchronization may occur within the space of several codewords, more quickly than in conventional coding schemes, and without the overheads implied by the regular periodic insertion of synchronization patterns [3]. Gilbert and Moore [2] noted that block codes can themselves never be self-synchronizing, but must rely on the introduction of coding mechanisms to achieve this. They showed that there exists an optimal length for the synchronizing prefix, for a given length block code. Comma-free codes [4], [5] introduce sufficient redundancy through the elimination of selected words, to enable the decoder to achieve correct synchronization within a finite bounded delay. Wei and Scholtz [6] develop the concept of a statistically synchronizable code, i.e., one where the synchronization delay is unbounded but for which the probability of synchronization approaches 1, in the limit. Others [7]–[11] propose variable-length code constructions which enable the decoder to correctly synchronize. Nevertheless, the range and richness of the structures implicit in variable-length codes renders the identification of general mechanisms for self-synchronization difficult, perhaps impossible.

Huffman’s construction [12] ensures optimal coding, i.e., minimal redundancy, but provides no assurance as to how the decoder will perform in the face of channel disturbances. In practice, errors may propagate for a considerable period, with an ensuing loss of many dozens of words, before synchronization is re-established. Ferguson and Rabinowitz [8] explored ways in which Huffman codes might be improved in respect of synchronization. The mechanism they use to explain self-synchronization, further developed in [9], relies on the existence of universal synchronizing codewords. These words are generally some of the longer codewords in the set capable of bringing about synchronization. However, in an optimally efficient code these are likely to occur infrequently, and may contribute in only a minor way to a set’s overall synchronization performance. Thus predictions based solely on the existence of universal synchronizing codewords may differ significantly from what is otherwise observed.

The thrust of the present paper is to model the self-synchronization process of the broader class of variable-length codes, using the results of the author’s work on the T-Codes [13]. T-Codes are variable-length codes produced from the recursive application of a set T-augmentation rule. These code sets are found to typically synchronize in just 2–3 codewords, using a mechanism which can be modeled directly in terms of the recursive T-augmentation construction. The basis for the code properties is well established [14], [15].

The T-Code synchronization model may be used to predict bounds on the behavior of a wide variety of code sets. The premise is that if two sets share a majority of their codewords, they will inevitably exhibit similar coding properties. The broad range of codes accessible within the T-Code class provides a basis for identifying similar sets, which share a large proportion of their elements. The differences which exist between these otherwise similar code sets, effectively emulate channel errors, as far as the T-Code synchronization process is concerned. The T-Code synchronization mechanism provides a means for determining an expected synchronization delay (ESD) which is in itself a good first-order estimate of other similar codes. By further understanding the contribution individual codewords have on the overall synchronization process, one may additionally infer probable bounds for the ESD of a given set.

Section II introduces notation. Section III provides a brief introduction to the extended T-Code construction. Section IV illustrates, by way of example, how the ESD may be determined for a T-Code set from the synchronization mechanism.
implicit in the set’s construction. Section V looks at two simple ways T-Code sets may be altered. Section VI applies this to specific examples, drawn from previous papers on the synchronization of variable-length codes.

II. PREREQUISITES

Denote by \( \mathbb{N} \) the set of natural numbers and let \( \mathbb{N}^+ = \mathbb{N} \setminus \{0\} \). Let the set \( A = \{a_1, \ldots, a_l\} \), where \( l \in \mathbb{N}, l \geq 1 \) be a finite alphabet. The elements of an alphabet \( A \) are referred to as symbols. The cardinality (number of elements) of a set \( A \) is denoted by \( \# A \). Hence \( \# A = l \). The set \( S^k \) denotes the free monoid generated by \( S \) under concatenation. The elements of \( A^* \) are called strings. The length of a string \( x \in A^* \) is denoted \( |x|_A \), or simply \( |x| \). \( \lambda \) denotes the empty string, \( |\lambda| = 0 \). A set \( S^+ \) is denoted by \( S^+ \). A finite set of strings \( S \subset A^+ \) is called a code set, and the elements of \( S \) are words, codewords or codestrings. For \( x, y, z \in S^+ \), if \( x = yz \) then \( y \) is a prefix of \( x \), denoted by \( y \preceq x \). Similarly, \( z \) is a suffix of \( x \), denoted by \( x \succeq z \). Given \( x \in S^+ \), the repeating pattern \( xx \cdots x \) is denoted \( x^n \), where \( n \) is the number of times \( x \) is repeated. For an alphabet \( R \) and code set \( S \subset A^* \), such that \( \# R = \# S \), then the one-to-one mapping between the elements of \( R \) and \( S \), denoted by the tuple \((R, S)\), will be called a code. The elements \( r \in R \) may have associated source probabilities \( P(r) \), such that

\[
\sum_{r \in R} P(r) = 1.
\]

Given a string \( x \in R^+ \), \( x = x_1 x_2 \cdots \), the encoding of \( x \), denoted \( \mathcal{E}_{(R,S)}(x) \), is a string \( y \in S^+ \), i.e., \( y = y_1 y_2 \cdots = \mathcal{E}(x) \), obtained from the consecutive substitution of symbols \( x_i \in R \) with corresponding strings \( y_i \in S \). If \( S \) is uniquely decipherable (UD), then the decoding of a string \( y \in S^+ \) is the string \( x \in R^+ \), denoted \( x = \mathcal{D}_{(R,S)}(y) \), obtained by the inverse substitution of elements. It follows that \( x = \mathcal{D}_{(R,S)}(\mathcal{E}_{(R,S)}(x)) \). This paper is concerned with prefix-free codes, i.e., codes which are both UD and instantaneous [16]. A prefix-free code \( S \) has the property that for all \( x, y \in S, x \preceq A \) implies \( x = y \). A code \((R, S)\) with source probabilities \( P(r) \) is said to be optimal with respect to the source probabilities, if

\[
\sum_{r \in R} P(r) |\mathcal{E}(r)|_A
\]

is minimal. For optimal codes we note the following approximate relation: \( P(r) \approx (\# A)^{-|\mathcal{E}(r)|} |\mathcal{E}(r)|_A \).

III. T-AUGMENTATION

For a code set \( S \), a string \( p \in S \) and positive integer \( k \in \mathbb{N}^+ \), the generalized T-augmentation of \( S \), denoted \( S^{(k)}(p) \), is given by

\[
S^{(k)}(p) = \{ p' u \mid i = 0, 1, \ldots, k ; u \in S\{p\} \cup \{p^{k+1}\} \}.
\]

The element \( p \) is referred to as the prefix element. The parameter \( k \) is referred to as an expansion parameter, since the size of the new code set derives from this. It is evident from (1), that all but one of the elements in \( S \), i.e., \((\# S - 1)\), are included in the new set \( S^{(k)}(p) \), \((k+1)\) times, with the prefixes \( p^0, p^1, \ldots, p^k \), respectively. These, together with the element \( p^{k+1} \), give

\[
\# S^{(k)}(p) = (k+1)(\# S - 1) + 1.
\]

From the standpoint of the decoder, any nonprefix element \( u \in S \setminus \{p\} \) encountered in the decoding of a string \( x = x_1 x_2 \cdots \), \( x_i \in S \) must, by virtue of (1), mark the interface between a codeword pair \( \cdots y_j y_{j+1} \cdots \), where \( y_j, y_{j+1} \in S^{(k)}(p) \). That is \( y_j \preceq u \). Thus any nonprefix element is a synchronizing pattern, automatically establishing the correct grouping of elements \( x_i \) and making the code \( S^{(k)}(p) \) manifestly self-synchronizing. Since \( p \in S \) appears as a prefix for elements of \( S^{(k)}(p) \), but also as a suffix in the element \( p^{k+1} \in S^{(k)}(p) \), it imparts no synchronization information to the decoder. It is necessary to retain \( p^{k+1} \) as an element to meet set completeness requirements. Since the probability of receiving the prefix \( p \) in an optimal code, is \( P(p) \approx (\# A)^{-|\mathcal{E}(p)|} |\mathcal{E}(p)|_A \), where \( \# A > 1 \), the probability of receiving a synchronizing pattern is favored, i.e., for \( |p| > 1 \) we have \((1 - P(p)) > 0.5 \).

A repeating occurrence of the prefix \( p \), i.e., \( p^k \), has the effect of delaying synchronization, yet where \( (l \gg k) \) the decoding of \( p^k \) will yield elements corresponding to \( p^{k+1} \) as in the intended encoding. In such situations, a decoding error may appear when synchronization is ultimately established. This is called an error-echo [15] since the initial error which brought about the loss of synchronization has occurred some time earlier. In between, information is correctly decoded, even though the decoder is not synchronized.

Applying the T-augmentation rule recursively \( n \) times, starting with an alphabet \( S \), gives rise to a set, denoted

\[
S_{(p_1, p_2, \ldots, p_n)}^{(k_1, k_2, \ldots, k_n)}
\]

or \( S_{\{p\}}^{(k_1, k_2, \ldots, k_n)} \) where \( p = (p_1, p_2, \ldots, p_n) \) and \( k = (k_1, k_2, \ldots, k_n) \), subject to the constraints \( k_i \in \mathbb{N}^+ \) and

\[
p_i \in S_{(p_1, p_2, \ldots, p_{i-1})}^{(k_1, k_2, \ldots, k_{i-1})}
\]

accordingly. Code sets formed from the recursive application of (1) are called T-Codes [17].

The size of a set \( S_{\{p\}}^{(k)} \) is a function of \( n, \# S \), and \( k \). Since the alphabet \( S \) is prefix-free, we have by induction from (2)

\[
\# S_{\{p\}}^{(k)} = 1 + (\# S - 1) \prod_{i=1}^{n} (k_i + 1).
\]

The recursive construction of \( S_{\{p\}}^{(k)} \) means that a decoder operating over the set may be viewed implicitly as simultaneously invoking a decoding over each and every one of the intermediate code sets

\[
S_{(p_1, p_2, \ldots, p_i)}^{(k_1, k_2, \ldots, k_i)} \quad \text{for } i = 1, \ldots, n - 1.
\]

Following an error, one may view the decoding as starting with respect to \( S \), and with synchronization being achieved for each of the intermediate sets \( S_{(p_1, p_2, \ldots, p_i)}^{(k_1, k_2, \ldots, k_i)} \) in turn, and finally with respect to \( S_{\{p\}}^{(k)} \) itself.

In \( n \) recursive applications of (1), it is broadly true that \( |p| \) increases as \( i \to n \). Therefore, the probability of receiving
a synchronizing pattern, \((1 - P(p_{C}))\), at increasing levels of 
T-augmentation, becomes increasingly favorable. Thus the T-
Codes have a rapid tendency to synchronize as a simple
consequence of the decoding process, independent of the
decoder implementation.

The cyclic repetition of any one, or more, of the prefix
patterns \(p_{C}\) will again have the effect of delaying synchronization.
The requirements for transitions to take place up through the
intermediate sets is formalized in [14] in terms of prefix- and
suffix-conditions. The synchronizing patterns, \(\text{sync}(S_{p}^{k})\), have the
property

\[
\text{sync}(S_{p}^{k}) \subset P_{1}^{k}P_{2}^{k} \cdots P_{n}^{k}(S_{p_{c}}^{(k_{1}, \ldots, k_{n-1})} \setminus \{p_{n}\}),
\]

(4)

However, not all patterns having the form on the right-hand
side of (4) are necessarily synchronizing strings.

IV. THE EXPECTED SYNCHRONIZATION DELAY (ESD)

A prefix-free code set \(S \subset A^{+}\) may be represented graphi-
cally as an \(I\)-ary branching tree (\(I \neq \# A\)). If every branching
node gives rise to precisely \(I\) branches then \(S\) is said to be
complete or exhaustive [2]. One may verify from (1), that if \(S\) is complete, then \(S_{p}^{k}\) is also complete. Since an alphabet is
complete by definition, the T-Codes are complete by induction.
The exclusion of any one element at an intermediate level
during T-augmentation, precludes the set-completeness property
from being satisfied at higher levels of T-augmentation.

For a code to be self-synchronizing, it must be capable
decoding any arbitrary string that could arise from an
error process. Thus set completeness is mandatory for a self-
synchronizing code. We are therefore interested in how T-Code
sets may be transposed into other complete code sets, and to
use what is understood of the T-Code synchronization process
to evaluate the expected synchronization performance of the
new code sets.

The modeling of the T-Code synchronization process is
covered in detail in [14]. Here the ESD for a T-Code set
is determined by way of a state machine representation for
the corresponding synchronization process. Transition proba-
bilities, \(P(0)\) and \(P(1)\), are assigned to each of the transition
paths. For an optimal code, we may assume that the probability
of receiving a “1” is about the same as for a “0,” i.e.,
\(P(1) \approx P(0)\). Under this premise we have noted that each
codeword \(x \in S_{p}^{k}\) of length \(|x|\) has a corresponding source
probability \(P(D(R_{S}(x))) \approx 1/(#S)^{|x|}\). It is further verifiable
that the decoder will, for an optimally encoded string, output
the codeword \(x\) with this same probability, irrespective of
whether the decoder is correctly synchronized. Thus the short
codewords dominate the statistics for the synchronization
process.

Next, the probability \(P_{m}\) of being in the synchronized state
may be calculated as a function of the number \(m\) of bits
received. The authors of [14] provide us with the result

\[
\text{ESD}(S_{p}^{k}) = \sum_{m=0}^{\infty} (1 - P_{m}),
\]

(5)

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\(S\) & \(S^{(1,1)}_{(0,1)}\) & \(S^{(4,1)}_{(0,4)}\) \\
\hline
0 & 0 & 0 \\
1 & 1 & 0 \\
10 & 10 & 00 \\
11 & 11 & 010 \\
& & 011 \\
\hline
\end{tabular}
\caption{Constructing \(S^{(1,1)}_{(0,1)}\) and corresponding level
transition diagram, where \(S = \{0, 1\}\).}
\end{table}

Example 1: Table I shows the construction of \(S^{(1,1)}_{(0,1)}\) where
\(S = \{0, 1\}\). A level transition-diagram is developed by linking
the intermediate code sets for each of the nonprefix code strings
which bring about synchronization at the respective levels. A
finite-state machine representation for this transition diagram
is depicted in Fig. 1. The paths are marked accordingly with
“0”s and “1”s corresponding to the input symbols which give
rise to the transitions. The decoder starts in the unsynchronized
state, and arrives in the synchronized state after making a series
of transitions reflecting the decoder input stream.

The probability \(P_{m}\) converges quickly to 1 (Fig. 2). Thus
the ESD for \(S^{(1,1)}_{(0,1)}\) is determined from relatively few terms, to
be ESD \((S^{(1,1)}_{(0,1)}) = 5.0\) bits. It seems that the synchronization
performance of a T-Code set generally is not a sensitive
function of the symbol probabilities. Varying the ratio \(P(0) : P(1)\)
(Fig. 3) confirms this for \(S^{(1,1)}_{(0,1)}\). Whereas, for \(P(0) = P(1) = 0.5\), ESD \((S^{(1,1)}_{(0,1)}) = 5.0\) bits, for \(P(0) = 1, P(1) = 0.6\)
the ESD rises only slightly to ESD \((S^{(1,1)}_{(0,1)}) = 5.2\) bits. Thus
an assumption that the code is optimal may be justifiable over a
range of situations.

We now explore how sensitive the ESD is to small mod-
fications to a set. Not all alterations to a set preserve set
completeness which we have observed, is mandatory for self-
synchronization. We consider three ways to modify a given
set. First, we have \textit{T-augmentation} itself to consider. Here, the T-Code synchronization model provides the framework for examining the shift in the ESD brought about by T-augmentation. T-augmentation is of especial interest since it selectively removes the transition path in the synchronization table corresponding to the prefix code. Thus T-augmentation may be used as a tool to estimate the effect of removing or modifying selected codewords.

Since the code’s behavior is dominated by the more frequently occurring codewords, i.e., the shorter strings, one may suppose that the “removal” of a short-prefix codeword would alter the ESD more than for a longer prefix code. Fig. 3 provides some assurance that shifts in the symbol probabilities brought about by modification to the set, have only a second-order effect on the ESD.

\textbf{Example 2:} Table II lists the elements of the fifth level T-augmented set \( S^{(1,1)}_{(0,1,0000,011)} \) where \( S = \{0,1\} \). The ESD(\( S^{(1,1,1,1,1)}_{(0,1,0000,011)} \)) as determined from the state machine for the synchronization process [17], is 10.82 bits. Table II further includes two of 33 sixth-level T-augmented sets possible for \( k_0 = 1 \), formed from \( S^{(1,1,1,1,1)}_{(0,1,0000,011)} \). The two sets represent extreme constructions in respect of the prefix lengths (3 bits and 9 bits respectively). Fig. 4 plots the ESD’s for all 33 of these sixth-level sets, as a function of their respective prefix lengths, revealing a clear trend. Since all 33 sets share all but one of their codewords with \( S^{(1,1,1,1,1,1)}_{(0,1,0000,011)} \), it is not unexpected that the ESD values fall into a relatively narrow band: 10.81–12.23 bits, close to ESD(\( S^{(1,1,1,1,1,1)}_{(0,1,0000,011)} \)) = 10.82 bits.

The trend indicates that, as a rule of thumb, each codeword contributes to the ESD in proportion to its frequency of occurrence. For example, the 3-bit-long codeword ”100” occurs one eighth of the time in an optimally coded data stream. Thus the removal of the 3-bit code effects an upward shift in the ESD by roughly 12.5\% \( \left( \frac{1}{8} \right) \). This rule, that a prefix effects an upward shift proportional to its probability of occurrence, provides a useful estimate of an upper bound (Fig. 4). Only two values fall beyond the estimated bound, and then only marginally so. However, the contribution of many codewords is more complex than that accounted for by the frequency of occurrence, as evident in the somewhat depressed range of ESD values shown.
For a few of the longer prefix selections the

$$\text{ESD}\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right)$$

even drops marginally below the

$$\text{ESD}\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right).$$

Most are above

$$\text{ESD}\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right) = 10.82 \text{ bits}$$

but well below the estimated upper bound. Almost all codewords include, to a greater or lesser extent, intermediate level prefixes which have the effect of delaying synchronization. This is especially evident in certain strings such as “1111” ($= 1^4 = (p_1)^4 = (11)^2 = (p_5)^2$). Thus the removal of this code produces an improvement (i.e., a reduction) in the ESD which almost offsets its delay contribution to the ESD at other levels.

Indeed,

$$\text{ESD}\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right) = 10.83 \text{ bits}$$

is very close to

$$\text{ESD}\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right) = 10.82 \text{ bits},$$

The largest decreases occur with the strings “110001” (comprising prefixes 1/1/00/01; ESD $\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right) = 10.80 \text{ bits}$) and “111000” (comprising prefixes 1/1/1/00; ESD $\left( S^{(1,1,1,1,1)}_{(0,1,0000,111)} \right) = 10.81 \text{ bits}$).

In summary then, an estimated upper bound for the ESD for the new set may be determined simply as a function of the length of the code string central to the modification in the set. Given a T-Code set $S^k_P$ and node $x \in S^k_P$, we have a useful estimate for the ESD for a set $S'$, derived from $S^k_P$ about $x$, denoted $S^k_P \xrightarrow{x} S'$

$$\text{upperbound} \left( \text{ESD}(S') \right) \approx \text{ESD}(S^k_P) \times (1 + \#S^{-\text{imp}}). \quad (6)$$

Similarly, we have

$$\text{lowerbound} \left( \text{ESD}(S^k_P) \right) \approx \text{ESD}(S'). \quad (7)$$

Alternatively, T-augmentation can provide a very specific prediction of the ESD resulting from changes to a T-code set involving a given codeword in the set.

V. NODE REDUCTION AND NODE EXTENSION

Two elementary steps may be used to transform any initial set to another final set, preserving set completeness. These may be used to transform T-Code sets to other sets. Referring back to the graphical representation for the code set, i.e., an inverted tree, (c.f., Fig. 5) leaf-node reduction (LN reduction) may be defined as the elimination of the $\#S$ leaf nodes stemming from a given branch node. Conversely, leaf-node extension (LN extension) may be defined as the populating of an existing leaf node with $\#S$ branches.

LN reduction and LN extension for a given $x \in S^k_P$ represent more modest alterations of the code tree than T-augmentation. In this respect, the change to the ESD brought about by T-augmentation provides a probable upper bound for the effects of LN reduction and LN extension. Nevertheless, LN reduction and LN extension may be applied successively and in combination to transform any initial set to any desired final set. The fewer the number of changes, and in particular, the longer the altered codewords, the “closer” the final set is to the initial set. By purposely restricting ourselves to small numbers of changes to an initial T-Code set, the synchronization model for the T-Code set will continue to apply, largely unaffected. Given an arbitrary variable-length code for which the ESD is to be evaluated, the challenge
becomes that of identifying a T-Code set which is "closest" to the initial set, in the sense mentioned above. Since the shorter codewords dominate the synchronization process, it is desirable that LN reduction and LN extension would use the longer rather than shorter codewords.

The T-Code synchronization model is then exactly applicable for periods during which any of the "altered" codewords do not appear, either in the encoding, or during decoding. The appearance of "altered" codewords during the synchronization process mimic timing errors at the decoder, with the effect of setting back the synchronization state and therefore increasing the ESD for the set relative to that for the corresponding T-Code set.

The basic approach to derive a probable bound on the ESD, is i) to use the state model for the T-Code synchronization process as an approximate method for deducing the ESD for the new set, and ii) to adjust the ESD by an amount corresponding to the shift expected from T-augmentation, for each of the altered codewords.

**Example 3:** Fig. 6 shows LN extension of the word (00) → 000 + 001. Using the T-augmentation process to provide an estimate of the ESD for the new set, we use the prefix "(00)" to create the next level set, $S^{(1,1,1)}_{(1,0,00)}$ for which the T-Code synchronization model gives

$$\text{ESD}(S^{(1,1,1)}_{(1,0,00)}) = 6.5 \text{ bits},$$

The ESD for the unmodified T-Code set $S^{(1,1,1)}_{(1,0,0)}$ represents a probable lower bound for the new set. We infer from this that the ESD for the new set shown to the right in Fig. 6 is in the vicinity of 5.0–6.5 bits.

**VI. SELF-SYNCHRONIZING HUFFMAN CODES**

Ferguson and Rabinowitz [8] note that Huffman codes have widely varying synchronization capabilities. They formulate a framework within which certain sets might be selected preferentially or modified to obtain improved synchronization performance. They observe, in example sets, that specific words, which they call universal synchronizing sequences, automatically establish synchronization irrespective of the decoder’s prior state. Implicit in their development is the assumption that sets which contain universal synchronizing sequences are to be preferred. This approach has a certain appeal, but Ferguson and Rabinowitz themselves conclude that the synchronization process must be more complex than they have supposed, simply because their calculations of the ESD often fall wide of the observed synchronization performance.
The T-Codes provide a possible explanation for this. In T-Code sets, short codewords appearing in combination with one another, are often equally capable of achieving synchronization as some of the longer words, but more likely to occur.

Ferguson and Rabinowitz introduce a classification for sets, synchronous and nonsynchronous, based on the existence of these universal synchronizing codewords. Maxted and Robinson [10] query the value of this classification, after finding examples of synchronous and nonsynchronous sets having similar tendencies for synchronization. These observations are confirmed here, although [10] uses quite a different synchronization measure from the one used here.

Table III tabulates two Huffman codes and introduced in [8] and given as examples of synchronous and nonsynchronous codes. C2, as it turns out, is the T-Code set introduced in Table I. The ESD for this set, calculated using the probabilities listed in Table III, is 5.07 bits, 1.5% larger than the earlier value calculated on the assumption \( P(1) = P(0) \) (Section III). None of the subsequent code sets discussed in [8] are T-Codes. Thus C2 may not be assumed automatically to be indicative of the behavior of their other examples in [8].

Ferguson and Rabinowitz identify D = "010" and E = "011" as universal synchronizing words for C2, thereby making the set a synchronous code. However, the transition table (Table I) shows that any string of the form

\[
1^{m_1}0^{m_2+1}[10]/[11], \quad m_1, m_2 \in \mathbb{N}
\]  

(8)

will bring about decoder synchronization. The solidus, /, in (8) indicates mandatory selection of either the "10" or "11" pattern. Precisely half of the synchronizing sequences, namely \(1^{m_1}0^{m_2+1}[10]/[11]\) with i) \(m_1 = 2m_1\) and \(m_2 = 2m_2\) or ii) \(m_1 = 2m_1 + 1\) and \(m_2 = 2m_2 + 1\), where \(m_1, m_2 \in \mathbb{N}\), decode with a terminating D or E. In this respect, D and E have the function of universal synchronizing words, though the T-Code model undergoes synchronization state transitions in advance of the occurrence of the D and E.

The remaining synchronizing sequences given by iii) \(m_1 = 2m_1 + 1\) and \(m_2 = 2m_2\) and iv) \(m_1 = 2m_1 + 1\) and \(m_2 = 2m_2 + 1\), where \(m_1, m_2 \in \mathbb{N}\), include neither D nor E, but terminate with B or C. Thus symbol combinations, for example, "0000010" (AAB), "1110010" (CBB), or "10111" (BC), are all valid synchronizing sequences unaccounted for in [8].

As Fig. 7 illustrates, synchronization is, for \(C_3\), achieved by letter combinations for 75% of all cases. The only single-character synchronizing sequences are "010" and "011," i.e., D and E, respectively, which account for some 25% of the synchronizing events. As observed above, half of all synchronizing patterns end with D or E. Only a third of all multiple-character synchronization patterns end in D or E. Clearly, a model of the synchronization mechanism which accounts only for events that include D and E will considerably underestimate the synchronization tendency for the set.

Two further examples (Table IV), also introduced in [8], may be explored in terms of LN reduction and LN extension. The first of these sets, \(C_3\), may be derived from LN extension of \(C_3\). From what has already been said about "closely related" sets, we expect \(s^{(1,1)}_{(0,1)}\) and \(C_3\) to have similar synchronization capabilities. Interestingly, [8] identifies \(C_3\) as a nonsynchronous code, devoid of universal synchronizing codewords.

The example deriving \(C_3\), using LN extension \((00 \rightarrow 000 + 001)\) on \(s^{(1,1)}_{(0,1)}\), has already been presented in Section V (Fig. 6). From a deterministic point of view, provided the encoded data stream does not include the letters C or D, and the T-Code decoder for \(C_3\) also does not output either of these letters during resynchronization, the mechanism for \(C_3\) is precisely that of the T-Code set, \(C_2\). Whereas, in Section V, \(00 \in s^{(1,1)}_{(0,1)}\) was assumed to occur in 25% of all cases, Table IV gives the combined probability for the pair of codewords \(000 + 001\), derived from the "00" at \(\approx 0.2\). Thus the ratio \(P(1): P(0)\) remains largely unchanged, and we may conclude from the calculation in Section V that the ESD \((C_3)\) will be in the vicinity of 5.0–6.4 bits.

Table V confirms the readiness of the nonsynchronous set \(C_3\) to resynchronize following any error, proving that it is
TABLE V
EVIDENCE OF SIGNIFICANT SYNCHRONIZATION POTENTIAL FOR THE NONSYNCHRONOUS SET C3

| Message | A | E | B | A | F | D | C | A | B | B | ...
|---------|---|---|---|---|---|---|---|---|---|---|---
| Encoded using C3 | 10.0 | 10.1 | 11.0 | 011.001.000.10.11.11...
| Missing bit in 3rd place | 0
| ... | 10.0 | 10.1 | 11.0 | 011.001.000.10.11.11...
| Decoded using C3 | A | A | B | A | F | D | C | A | B | B | ...
| Inverted bit in 3rd place | 1
| ... | 10.0 | 10.1 | 11.0 | 011.001.000.10.11.11...
| Added bit in 4th place | 1
| ... | 10.0 | 10.1 | 11.0 | 011.001.000.10.11.11...

at least statistically synchronizable. In this respect the label nonsynchronous could be considered misleading.

In the case of C4, Ferguson and Rabinowitz identify $F = "101" \in C_4$ as a universal synchronizing word. Thus C4 is a synchronous code set. Were the label indicative of relative performance, C4 might be assumed to be significantly better than C3 in respect of synchronization. A search of T-Code sets finds $S^{(2.1)}_{(0.1)}$ (Table VI) to be “close” to C4. LN reduction ($1000 + 1001 \rightarrow 100$) of $S^{(2.1)}_{(0.1)}$ results in C4 (Table VII).

The synchronizing sequences for the T-Code set, $S^{(2.1)}_{(0.1)}$, are directly applicable to C4, subject to the appearance of the altered codeword “100,” corresponding to the letter E. From Table VI the synchronizing sequences for $S^{(2.1)}_{(0.1)}$ are of the general form

$$0^{m_1}1^{m_2}+1[0^2]/[1^2]/[0^2]1^1, \text{ where } m_1, m_2 \in \mathbb{N}. \tag{9}$$

Setting $m_1 = m_2 = 0$ in (9) confirms that $F = "101"$ has the effect of synchronizing the decoder. However, the additional options $[0^2]/[1^2]/[0^2]$ give rise to an infinity of bit sequences which do not include the letter $F$, yet contribute in a significant fashion to the overall synchronization capability of the set.

The ESD ($S^{(2.1)}_{(0.1)}$) = 5.5 bits provides an estimated lower bound for ESD ($C_4$). The contribution that “1000” and “1001” make to the synchronization delay is small since these are the longer codewords in the set. T-augmentation of $S^{(2.1)}_{(0.1)}$ using prefixes “1000,” and “1001”, indeed produces no change to the ESD. Since the altered code “100” corresponding to E occurs some 10% of the time, the ESD ($C_4$), using our rule of thumb, can be expected to move upwards by no more than 5.5 x 0.1 = 0.55 bits, i.e., to 6.05 bits. Thus the difference between ESD ($C_3$) and ESD ($C_4$) is almost certainly < 1 bit.

Maxted and Robinson [10] also evaluated $C_3$ and $C_4$, though using a somewhat different measure of synchronization performance, an average character error span, denoted $E_s$. Their analysis assumed slightly different letter probabilities, but from what has already been said, the synchronization performance will not strongly reflect this. They concluded that there is no difference between the two sets, having determined that each has an average character error-span, $E_s = 1.2$ characters. They further observed that “the existence of a synchronizing codeword may be sufficient to guarantee statistical synchronization but is not necessary” (author’s italics).

VII. CONCLUSION

The present paper has demonstrated that the synchronization mechanism for the T-Codes may be used to model the synchronization behavior of a range of other variable-length codes. T-augmentation may be used very specifically to estimate the contribution individual words make to the ESD for the set. Whereas a closely related T-Code set provides a good first-order estimate of the ESD for the set, T-augmentation may additionally be used to derive probable bounds on the ESD for the set, taking into account the individual codeword differences between the T-Code set and the initial code set. The T-Code synchronization mechanism derives from the recursive T-augmentation construction. Thus it is readily accessible and generally straightforward to evaluate.

Two further methods are used for deriving “close” sets from T-Code sets. It is argued that the effects of simple leaf-node extension and leaf-node reduction, which preserve set completeness, are smaller than those brought about by T-augmentation using corresponding leaf-nodes. The alteration of codewords by LN extension and LN reduction effectively

TABLE VI
CONSTRUCTION OF THE SET $S^{(2.1)}_{(0.1)}$ WHERE $S = \{0, 1\}$

<table>
<thead>
<tr>
<th>S</th>
<th>S^{(2.1)}_{(0.1)}</th>
<th>S^{(2.1)}_{(0.3)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>1001</td>
<td>1001</td>
<td>1001</td>
</tr>
</tbody>
</table>

TABLE VII
TRANSFORMATION OF $S^{(2.1)}_{(0.1)}$ BY NODE REDUCTION:

$1000 + 1001 \rightarrow 100$

<table>
<thead>
<tr>
<th>$S^{(2.1)}_{(0.3)}$</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>1000</td>
<td>100</td>
</tr>
<tr>
<td>1001</td>
<td>100</td>
</tr>
</tbody>
</table>
emulates timing errors, as far as the T-Code synchronization model is concerned, having the effect of setting back the synchronization process.

Examples introduced in the paper seem to suggest that a classification of sets based on the existence of universal synchronizing codewords is not necessarily indicative of the actual synchronization performance of a code, a point also noted by Maxted and Robinson. It is further interesting to note that the character span values calculated by Maxted and Robinson, when translated into bits, fall substantially below the ESD values predicted by the T-Code models. A possible explanation for this might be that the T-Code model assumes a decoder’s perspective. That is, having no a priori knowledge of the message being received, it is not possible to determine whether synchronization has been established by chance at an earlier position. Thus an observer who has available both the input and output data stream will, for a significant number of situations, conclude that the output stream is synchronized before the point when a T-Code decoder can conclude this.

ACKNOWLEDGMENT

The author wishes to thank U. Günther for his helpful comments and attention to detail, during the final preparation of this paper.

REFERENCES