ISIT 2003, Yokohama, Japan, June 29 – July 4, 2003

Self-Synchronization of Huffman Codes[1]

Christopher F. Freiling
Department of Mathematics
California State University
San Bernadino, CA 92407
cfreilin@csusb.edu

Douglas S. Jungreis
Center for Commun. Research
4320 Westerra Court
San Diego CA 92121
jungreis@ccrwest.org

François Théberge
Dept. of National Defense
CSE/DND, P.O. Box 9703
Ottawa, K1G 3Z4, Canada
f_theberge@hotmail.com

Kenneth Zeger
Dept. of Electrical & Comp. Eng.
Univ. of California, San Diego
La Jolla, CA 92093-0407
zeger@ucsd.edu

Variable length binary codes have been frequently used for communications since Huffman's important paper on constructing minimum average length codes. One drawback of variable length codes is the potential loss of synchronization in the presence of channel errors. However, many variable length codes seem to possess a "self-synchronization" property that lets them recover from bit errors.

In particular, for some variable length codes there exists a certain binary string (not necessarily a codeword) which automatically resynchronizes the code. That is, if a transmitted sequence of bits is corrupted by one or more bit errors, then as soon as the receiver by random chance correctly detects a self-synchronizing string, the receiver can continue properly parsing the bit sequence into codewords. Most commonly used binary prefix codes, including Huffman codes, are "complete", in the sense that the vertices in their decoding trees are either leaves or have two children. An open question has been to characterize which prefix codes and which complete prefix codes have a self-synchronizing string. In this paper, we prove that almost all complete prefix codes have a self-synchronizing string.

Capocelli, Gargano, and Vaccaro [1] proved that a variable-length code is statistically synchronizable iff it has a self-synchronizing string. They also gave an algorithm that determines whether a code has a self-synchronizing string. Capocelli, et al. [2] gave an algorithm for constructing prefix codes with self-synchronizing strings such that the average length of the code is close to optimal. They also provided a method for constructing prefix codes with a self-synchronizing codeword and whose rate redundancy is low.

Ferguson and Rabinowitz [3] found sufficient conditions for the existence or non-existence of self-synchronizing strings for Huffman codes for many classes of source probabilities, but required the synchronizing strings to be Huffman codewords. They also examined the problem of finding, for a given set of codeword lengths, a Huffman code with the shortest possible self-synchronizing codeword. Montgomery and Abrahams [4] showed how to construct variable length codes with a self-synchronizing string, whose average length is close to that of a Huffman code.

A prefix code is *complete* if for every $u \in \{0,1\}^*$, the string $u0$ is a prefix of some codeword if and only if $u1$ is a prefix of some codeword. Huffman codes are examples of complete prefix codes.

A *binary tree* is a finite directed acyclic graph such that every node has out-degree zero or two, one node (called the *root node* and denoted r) has in-degree zero, and all other nodes have in-degree one. Whenever an edge leads from one node to another, these nodes are referred to respectively as the *parent* and *child*. The edges leading from a parent to its children are labeled '0' and '1', and the corresponding children are called the *0-child* and *1-child*. A *leaf* is a node with no children. A non-leaf node is called *internal*. A *branch* is a path from the root to a leaf, and each branch is identified with the sequence of zeros and ones that label this path. In particular, the *zero branch* is the branch associated with the all zeros codeword. Each node of a tree is identified with the label of the path to the node from the root. In particular, the root node is identified with the empty string.

A complete prefix code can be conveniently represented by the binary tree whose branches are its codewords. To decode a binary sequence, one places a pointer at the root, and proceeds through the bit sequence. For each 0-bit, the pointer is moved to its 0-child, and for each 1-bit the pointer is moved to its 1-child. Whenever the pointer reaches a leaf, the symbol in $\mathcal{A}$ that is represented by that leaf's path is output, and the pointer is reset to the root.

A simple but useful fact is that for every node v in a binary tree, there exists a nonnegative integer k such that $v0^k$ is the root node, where 0^k is the string of k zeros. This is because we can traverse down a binary tree from v along 0-children, eventually hitting a leaf node.

Recall that in the decoding procedure, we keep a pointer to a node in the tree; for each bit of data, we move the pointer from a node to one of its children, and when we reach a leaf, we move the pointer back to the root. In this way, any string of data-bits moves the pointer from one node to another. We define this formally:

In the decoding procedure, the location of the pointer depends on the entire set of bits that have been decoded; however, knowing the most recently decoded bits provides some information about the location of the pointer. In particular, there are certain strings that bring multiple nodes to one node.

A string $z \in \{0,1\}^*$ is *self-synchronizing* for a complete prefix code if it brings every internal node to the root. Therefore, if we are decoding a data stream and we have just decoded a self-synchronizing string, then the pointer is back at the root, regardless of what bits preceded the self-synchronizing string.

Let $\mathcal{Q}$ be a collection of binary codes and let P be some property that each such code may or may not have. For each positive integer n let $\alpha(n)$ be the number of codes in $\mathcal{Q}$ with n codewords that have property P, and let $\beta(n)$ be the number of codes in $\mathcal{Q}$ with n codewords. We say that *almost all codes in $\mathcal{Q}$ have property P* if $\lim_{n\to\infty} \alpha(n)/\beta(n) = 1$. The main result of this paper is the following theorem.

Theorem 1 *Almost all complete prefix codes have a self-synchronizing string.*

References

[1] R. M. Capocelli, L. Gargano, and U. Vaccaro. On the characterization of statistically synchronizable variable-length codes. *IEEE Transactions on Information Theory*, 34(4):817–825, July 1988.

[2] R. M. Capocelli, A. A. De Santis, L. Gargano, and U. Vaccaro. On the construction of statistically synchronizable codes. *IEEE Transactions on Information Theory*, 38(2):407–414, March 1992.

[3] T. J. Ferguson and J. H. Rabinowitz. Self-synchronizing Huffman codes. *IEEE Transactions on Information Theory*, 30(4):687–693, July 1984.

[4] B. L. Montgomery and J. Abrahams. Synchronization of binary source codes. *IEEE Transactions on Information Theory*, 32(6):849–854, November 1986.

[1]This research was supported in part by the National Science Foundation.