Optional Problem: Kraft inequality and full trees

(notes by Michael Baer)

This handout illustrates that the meaning of the Kraft inequality for infinite alphabet codes is not necessarily what we might expect.

A valid binary prefix code with a full encoding tree is one for which every binary string is the prefix of a valid codeword, contains a valid codeword as a prefix, or is a valid codeword itself. This is equivalent to every node in the code tree having two or zero children.

For finite alphabets, such a code must satisfy the Kraft inequality with equality. Consider a code. We can consider each codeword as occupying a portion of the $[0,1)$ interval beginning at its binary fractional expansion and occupying a subinterval of length $2^{-\ell}$, as in Figure 1. Thus, for example, the codeword 101 would occupy $[0.625, 0.75)$. It is clear that these intervals are disjoint for codewords of a prefix code and that their summed length is the Kraft inequality sum.

To show a finite code with a full tree must satisfy Kraft with equality, suppose there is a code that does not, one for which the Kraft sum is less than 1. (It cannot be greater as no coding tree could violate the Kraft inequality.) Thus there must be at least one empty gap lying between two intervals. Such a gap must contain an interval beginning at a point corresponding to another finite-length binary fractional expansion, ending at a point some (negative) power of two greater in value. This corresponds to a codeword that is not the prefix of a valid codeword, does not contain a valid codeword as a prefix, and is not valid codeword itself. Thus the code tree cannot be full. $\diamondsuit$

Consider the following code C:

$$
\begin{array}{lcllll}
c_1 & = & 0\ 0 & & & \\
c_2 & = & 0\ 1 & 0\ 0\ 0 & & \\
c_3 & = & 1\ 0 & 0\ 0\ 0 & & \\
c_4 & = & 1\ 1 & 0\ 0\ 0 & & \\
c_5 & = & 0\ 1 & 0\ 0\ 1 & 0\ 0\ 0\ 0 & \\
c_6 & = & 0\ 1 & 0\ 1\ 0 & 0\ 0\ 0\ 0 & \\
 & & & \ \ \ \vdots & &
\end{array}
$$

This code has infinite number of codewords, each comprised of blocks of increasing size. If a block consists of all zero bits, this indicates the codeword's termination. Otherwise, there is at least one more block in the codeword. Thus it is obviously a prefix code. However, we will show that, although corresponding to a full coding tree, it does not satisfy the Kraft inequality with equality.

We can formalize the code specification as was done in a paper by Linder, Tarokh, and Zeger ("Existence of Optimal Prefix Codes for Infinite Source Alphabets," *IEEE Transactions on*

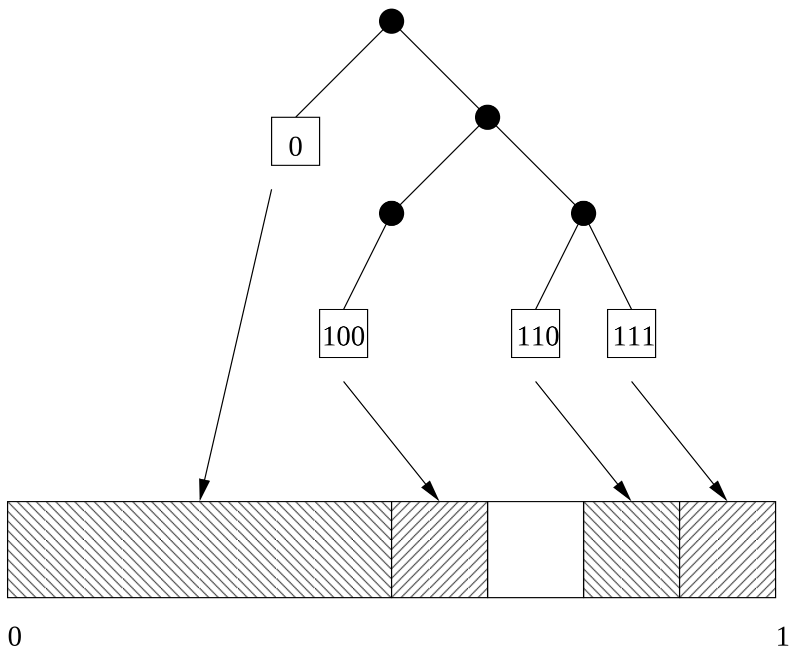

Figure 1: Representations of a prefix code as a tree (not full) and as subintervals of the unit interval

Information Theory, Vol. IT-43, pp. 2026-2028, 1997). Let $\mathrm{CAT}(a_1, a_2)$ and the equivalent $\mathrm{CAT}_{i=1}^{2} a_i$ be two methods of denoting binary concatenation. Let N_j denote the set of j-bit binary strings excluding 0^j, the j-bit binary string consisting of all 0's. Thus the size of the set N_j is $\mathrm{NUM}(N_j) = 2^j - 1$. We may now formally define:

$$C = \{00\} \cup \left[\bigcup_{k=2}^{\infty} \mathrm{CAT} \left\{ \left(\mathop{\mathrm{CAT}}_{j=2}^{k} N_j \right), 0^{k+1} \right\} \right].$$

If the tree were not full, there would exist a finite length sequence which is not the prefix of a valid codeword, does not contain a valid codeword as a prefix, and is not valid codeword itself, i.e. a sequence not compatible with the code. Suppose there exists such a sequence of length m. Note that we may pad the end of such a sequence with additional bits, because if the original sequence is incompatible, no compatible sequence contains it as a prefix. Let $k = \lceil \frac{1}{2}(-3 + \sqrt{9 + 8m}) \rceil$. This is the number of variable length blocks necessary to form the smallest codeword at least as long as this sequence, the first block being of length 2, the second of block 3, etc. We may pad the sequence with $m - \frac{1}{2}(k+1)(k+2) - 1$ zeroes, so it may be composed of such blocks.

Now we may show by induction that all possible sequences (of blocks) are compatible: It is obvious that one-block (two-bit) sequences are all valid codewordss ($\{00\}$) or prefixes to valid codewords ($\{01000, 10000, 11000\}$). Assume all sequences with $k-1$ such block are prefixes for valid codewords (those for which the next k digits are 0), are valid codewords, or have valid codewords as prefixes. In the last two cases, this remains true for the k-block sequence. Thus we may assume that the first $k-1$ blocks comprise the prefix to a valid codeword. If the next k digits are 0, we have a valid codeword. Otherwise, we have a prefix to a valid codeword made by concatenating the (padded) sequence with 0^{k+1} ($k+1$ additional 0's). Thus no such incompatible sequence exists, and the tree is full. $\diamondsuit$

Note that, although it is difficult to calculate the exact value of the Kraft inequality sum for this code, one may upper bound it as follows:

$$\begin{aligned}
\sum_i 2^{-\ell_i} &\overset{(a)}{=} \frac{1}{4} + \sum_{k=2}^{\infty} \mathrm{NUM}\left[\mathrm{CAT}\left\{\left(\operatorname*{CAT}_{j=2}^{k} N_j\right), 0^{k+1}\right\}\right]\left(2^{-\sum_{n=2}^{k+1} n}\right) \\
&\overset{(b)}{=} \frac{1}{4} + \sum_{k=2}^{\infty}\left(\prod_{j=2}^{k}(2^j-1)\right)\left(2^{-\sum_{n=2}^{k+1} n}\right) \\
&\overset{(c)}{<} \frac{1}{4} + \sum_{k=2}^{\infty}\left(2^{\sum_{n=2}^{k} n}\right)\left(2^{-\sum_{n=2}^{k+1} n}\right) \\
&\overset{(d)}{=} \sum_{k=1}^{\infty} 2^{-(k+1)} \\
&\overset{(e)}{=} \frac{1}{2}
\end{aligned}$$

Step (a) finds an expression for the Kraft inequality sum by multiplying the number of codewords of each possible length (length being equal to $\sum_{n=2}^{k+1} n$ for a k-block codeword) by their Kraft inequality term ($2^{-\ell_i} = 2^{-\sum_{n=2}^{k+1} n}$). Step (b) calculates explicitly the number of codewords of each length. Step (c) increases each multiplicative term to the smaller power of two greater than or equal to it. Step (d) cancels redundant terms. Step (e) calculates an infinite series.

The Kraft inequality is not satisfied with equality because each valid codeword of length m is required to end in $\frac{1}{2}(-1+\sqrt{9+8m})$ zeroes. Thus, although full, this code may be considered wasteful. Note that we may also use the analogy of Figure 1 to explain this. Because we have an infinite number of codewords, it is possible to fill in the $[0,1)$ interval in such a way as to leave no gaps and yet have the filling intervals sum to a number less than 1. $\diamondsuit$