LECTURE 10

CODING THEORY - I

Having looked at computers and how they operate, we now turn to the problem of the representation of information - how do we represent the information we want to process. Recall that any meaning that a symbol may have depends on how it is processed; there is no inherent meaning to the bits that the machine uses. In the synthetic language mentioned in Lecture 4 on the history of software, the breaking up of the instructions was pretty much the same for every code instruction and this is true for most languages; the "meaning" of any instruction is defined by the corresponding subroutine.

To simplify the problem of the representation of information we will, at present, examine only the problem of the transmission of information from here to there. This is exactly the same as transmission from now to then, storage. Transmission through time or through space are the same problem. The standard model of the system is given in Figure 10-1.

Starting of the left hand side of Figure 10.1 we have a source of information. We do not discuss what the source is. It may be a string of: alphabetical symbols, numbers, mathematical formulas, musical notes of a score, the symbols now used to represent dance movements - what ever the source is and what ever "meaning" is associated with the symbols is not part of the theory. We postulate only a source of information, and by doing only that, and no more, we have a powerful, general theory that can be widely applicable. It is the abstraction from details that gives the breadth of application.

When in the late 40's C. E. Shannon created Information Theory there was a general belief that he should call it Communication Theory, but he insisted on the word "information", and it is exactly that word which has been the constant source of both interest and of disappointment in the theory. One wants to have a theory of "information" but it is simply a theory of strings of symbols. Again, all we suppose is there is such a source, and we are going to encode it for transmission.

The encoder is broken into two parts, the first half is called the source encoding which as its name implies is adapted to the source, various sources having possibly different kinds encodings.

The second half of the encoding process is called channel encoding and it is adapted to the channel over which the encoded symbols are to be sent. Thus the second half of the encoding process is tuned to the channel. In this fashion, with the common interface, we can have a wide variety of sources encoded

first to the common interface, and then the message is further encoded to adapt it to the particular channel being used.

Next, going to the right in Figure 10.1, the channel is supposed to have "random noise added". All the noise in the system is incorporated here. It is assumed that the encoder can uniquely recognize the incoming symbols without any error, and it will be assumed that the decoder similarly functions without error. These are idealizations, but for many practical purposes they are close to reality.

Next, the decoding is done in two stages, channel to standard, and then standard to the source code. Finally it is sent on to the sink, to its destination. Again, we do not ask what the sink does with it.

As stated before, the system resembles transmission, for example a telephone message from me to you, radio, or TV programs, and other things such as a number in a register of a computer being sent to another place. Recall, again, that sending through space is the same as sending through time, namely storage. If you have information and want it later, you encode it for storage and store it. Later when you want it it is decoded. Among encoding systems is the identity, no change in the representation.

The fundamental difference between this kind of a theory and the usual theory in physics is the assumption at the start that there is "noise", that errors will arise in any equipment. Even in quantum mechanics the noise appears at a later stage as an uncertainty principle, not as an initial assumption; and in any case the "noise" in Information Theory is not at all the same as the uncertainty in Q.M.

We will, for convenience only, assume that we are using the binary form for the representation in the system. Other forms can be similarly handled, but the generality is not worth the extra notation.

We begin by assuming that the coded symbols we use are of variable length, much as the classical Morse code of dots and dashes, where the common letters are short and the rare ones are long. This produces an efficiency in the code, but it should be noted that Morse code is a ternary code, not binary, since there are spaces as well as dots and dashes. If all the code symbols are of the same length we will call it a block code.

The first obvious property we want is the ability to uniquely decode a message if there is no noise added - at least it seems to be a desirable property, though in some situations it could be ignored to a small extent. What is sent is a stream of symbols which looks to the receiver like a string of 0's and 1's. We call two adjacent symbols a second extension, three a third extension, and in general if we send n symbols the receiver sees the n-th extension of the basic code symbols. Not knowing n, you the receiver, must break the stream up into units that can be translated, and you want, as we said above, to be able at the

receiving end, meaning you again, to make this decomposition of the stream uniquely in order to recover the original message that I, at the sending end, sent to you.

I will use small alphabets of symbols to be encoded for illustrations; usually the alphabet is much larger. Typically natural language alphabets run from 16 to 36 letters, both upper and lower case, along with numbers and numerous punctuation symbols. For example, ASCII has $128 = 2^7$ symbols in its alphabet.

Let us examine one special code of four symbols, s_1, s_2, s_3. s_4.

$$s_1 = 0$$
$$s_2 = 00$$
$$s_3 = 01$$
$$s_4 = 11$$

If you receive

0011

what will you do? Is it

$s_1\ s_1\ s_4$ or is it $s_2\ s_4$?

You cannot tell; the code is not uniquely decodable, and hence is unsatisfactory. On the other hand the code

$$s_1 = 0$$
$$s_2 = 10$$
$$s_3 = 110$$
$$s_4 = 111$$

is uniquely decodable. Let us take a random string and see what you would do to decode it. You would construct a <u>decoding tree</u> of the form shown in Fig 10-2. The string

11010010011011100010100110...

can be broken up into the symbols

110,10,0,10,0,110,111,0,0,0,10,10,0,110, ...

by merely following the decoding tree using the rule:

Each time you come to a branch point (node) you read the next symbol, and when you come to a leaf of the tree you emit the corresponding symbol and return to the start.

The reason why this tree can exist is that no symbol is the prefix of any other, so you always know when you have come to the end of the current symbol.

There are several things to note. First, the decoding is a straight forward process in which each digit is examined only

once. Second, in practice you usually include a symbol that is an exit from the decoding process which is needed at the end of message. Failure to allow for an escape symbol is a common error in the design of codes. You may, of course, never expect to exit from a decoding mode, in which case the exit symbol is not needed.

The next topic is instantaneous decodable codes. To see what this is, consider the above code with the digits reversed end for end.

$$s_1 = 0$$
$$s_2 = 01$$
$$s_3 = 011$$
$$s_4 = 111$$

Now consider receiving 011111 ... 111. The only way you can decode this is to start at the final end and group by three's until you see how many 1's are left to go with the first 0; only then you can decode the first symbol. Yes, it is uniquely decodable, but not instantaneously! You have to wait until you get to the end of the message before you can start the decoding process! It will turn out (McMillan's Theorem) that instantaneous decodability costs nothing in practice, hence we will stick to instantaneously uniquely decodable codes.

We now turn to two examples of encoding the same symbols, s_i.

$$s_1 = 0$$
$$s_2 = 10$$
$$s_3 = 110$$
$$s_4 = 1110$$
$$s_5 = 1111$$

which will have the decoding tree shown in Figure 10-3.

The second encoding is the same source, but we have

$$s_1 = 00$$
$$s_2 = 01$$
$$s_3 = 10$$
$$s_4 = 110$$
$$s_5 = 111$$

with the tree shown in Figure 10-4.

The most obvious measure of "goodness" of a code is its average length for some ensemble of messages. For this we need to compute the code length l_i of each symbol multiplied by its corresponding probability p_i of occurring, and then add these products over the whole code. Thus the formula for the average code length L is, for an alphabet of q symbols,

$$L = \text{SUM}[i=1,q;\ p_i l_i]$$

where the p_i are the probabilities of the symbols s_i and the l_i are the corresponding lengths of the encoded symbols. For an efficient code this number L should be as small as possible. If $p_1 = 1/2$, $p_2 = 1/4$, $p_3 = 1/8$, $p_4 = 1/16$, and $p_5 = 1/16$, then for code #1 we get

$$L = 1(1/2) + 2(1/4) + 3(1/8) + 4(1/16 + 1/16) = 1\ 7/8$$

and for code #2

$$L = 2(1/2) + 2(1/4 + 1/8) + 3(1/16 + 1/16) = 2\ 1/8$$

and hence the given probabilities will favor the first code.

If most of the code words are of the same probability of occurring then the second encoding will have a smaller average code length that the first encoding. Let all the $p_i = 1/5$. The the code #1 has

$$L = (1/5)(1 + 2 + 3 + 4 + 4) = 14/5 = 2\ 4/5$$

while code 2 has

$$L = (1/5)(2 + 2 + 2 + 3 + 3) = 12/5 = 2\ 2/5$$

thus favoring the second code. Clearly the designing of a "good" code must depend on the frequencies of the symbols occurring.

We now turn to the Kraft inequality which gives a limit on the lengths l_i of the code symbols of a code. In the base 2, the Kraft inequality is

$$K = SUM[i=1,q;\ 1/2^{l}i] \leq 1$$

When examined closely this inequality says that there cannot be too many short symbols or else the sum will be too large.

To prove the Kraft inequality for any instantaneously uniquely decodable code we simply draw the decoding tree, which of course exists, and apply mathematical induction. If the tree has one or two leaves as shown in Figure 10-5 then there is no doubt that the inequality is true. Next, if there are more than two leaves we decompose the trees of length m (for the induction step) into two trees, and by the induction suppose that the inequality applies to each branch of length m-1 or less. By induction the inequality applies to each branch, giving K′ and K" for their sums. Now when we join the two trees each length increases by 1, hence each term in the sum gets another factor of 2 in the denominator, and we have

$$(1/2)K' + (1/2)K'' \leq 1$$

and the theorem is proved.

Next we consider the proof of McMillan's Theorem that the Kraft inequality applies to non-instantaneous codes provided they

are uniquely decodable. The proof depends on the fact that for any number K > 1 some n-th power will exceed any linear function of n, when n is made large enough. We start with the Kraft inequality raised to the n-th power (which gives the n-th extension) and expand the sum

$$K^n = \text{SUM}[i=1,q;\ 1/2^{l_i}]^n = \text{SUM}[k=n,nl;\ N_k/2^k]$$

where N_k is the number of symbols of length k, and the sum starts from the minimum length of the n-th extension of the symbols, which is n, and ends with the maximum length nl, where l is the maximum length of any single code symbol. But from the unique decodability it must be that $N_k \leq 2^k$. The sum becomes

$$K^n \leq \text{SUM}[k=n,nl;\ 2^k/2^k] = nl - n + 1$$

If K were > 1 then we could find an n so large that the inequality would be false, hence we see that $K \leq 1$, and McMillan's Theorem is proved.

Since we now see, as we said we would show, that instantaneous decodability costs us nothing, we will stick to them and ignore merely uniquely decodable codes - their generality buys us nothing.

Let us take a few examples to illustrate the Kraft inequality. Can there exist a uniquely decodable code with lengths 1, 3, 3, 3? Yes, since

$$1/2 + 1/8 + 1/8 + 1/8 = 7/8 < 1$$

How about lengths 1, 2, 2, 3? We have

$$1/2 + 1/4 + 1/4 + 1/8 = 9/8 > 1$$

hence no! There are too many short lengths.

Comma codes are codes where each symbol is a string of 1's followed by a 0, except the last symbol which is all 1's. As a special case we have

$s_1 = 0$
$s_2 = 10$
$s_3 = 110$
$s_4 = 1110$
$s_5 = 1111$

We have the Kraft sum

$$1/2 + 1/4 + 1/8 + 1/16 + 1/16 = 1$$

and we have exactly met the condition. It is easy to see that the general comma code meets the Kraft inequality with exact equality.

If the Kraft sum is less than 1 then there is excess signal-

ing capacity since another symbol could be included, or some existing one shortened and thus the average code length would be less.

Note that if the Kraft inequality is met that does not mean that the code is uniquely decodable, only that there exists a code with those symbol lengths that is uniquely decodable. If you assign binary numbers in numerical order, each having the right length l_i in bits, then you will find a uniquely decodable code. For example, given the lengths 2, 2, 3, 3, 4, 4, 4, 4 we have for Kraft's inequality

$$2(1/4) + 2(1/8) + 4(1/16) = 1$$

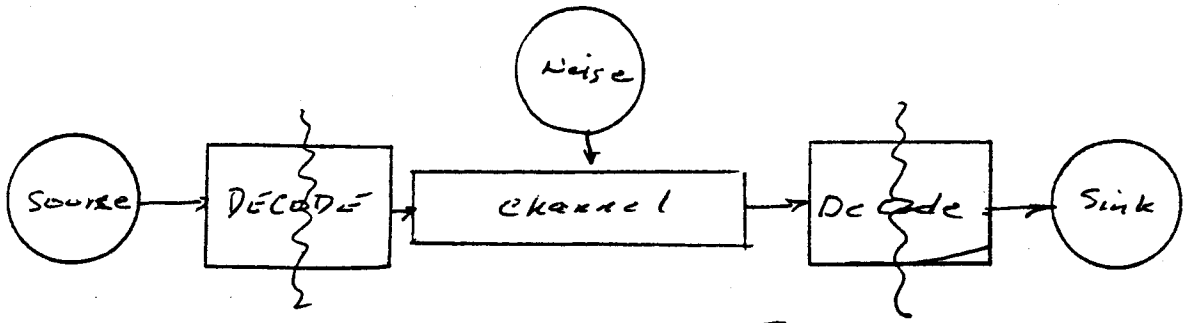
hence an instantaneously decodable code can exist. We pick the symbols in increasing order of numerical size, with the binary point on imagined on the left, as follows, and watch carefully the corresponding lengths l_i:

$s_1 = 00$
$s_2 = 01$
$s_3 = 100$
$s_4 = 101$
$s_5 = 1100$
$s_6 = 1101$
$s_7 = 1110$
$s_8 = 1111$

I feel it necessary to point out how things are actually done by us when we communicate ideas. Thus I want, at this time, to get an idea from my head into yours. I emit some words from which you are supposed to get the idea. But if you later try to transmit this idea to a friend you will emit, almost certainly, different words. In a real sense, the "meaning" is not contained in the specific words I use since you will probably use different words to communicate the same idea. Apparently different words can convey the same "information". But if you say you do not understand the message then usually a different set of words is used by the source in a second or even third presentation of the idea. Thus, again in some sense, the "meaning" is not contained in the actual words I use, but you supply a great deal of surrounding information when you make the translation from my words to your idea of what I said inside you head.

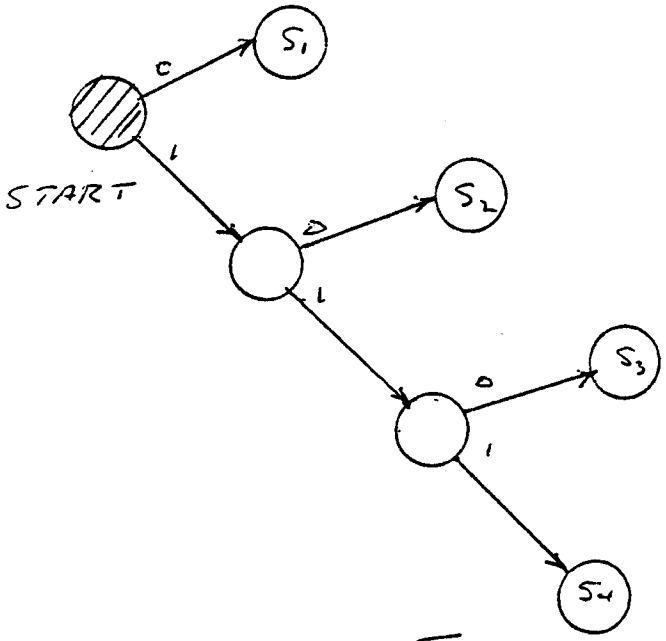
We have learned to "tune" the words we use to fit the person on the receiving end; we to some extent select according to what we think is the channel noise, though clearly this does not match the model I am using above since there is significant noise in the decoding process, shall we say. This inability of the receiver to "hear what is said" by a person in a higher management position but to hear only what they expect to hear, is, of course, a serious problem in every large organization, and is something you should be keenly aware of as you rise towards the top of the organization. Thus the representation of information in the formal theory we have given is mirrored only partly in life as we live it, but it does show a fair degree of relevance

outside the formal bounds of computer usage where it is highly applicable.



Information System

Figure 10-1



Decoding Tree

Figure 10-2

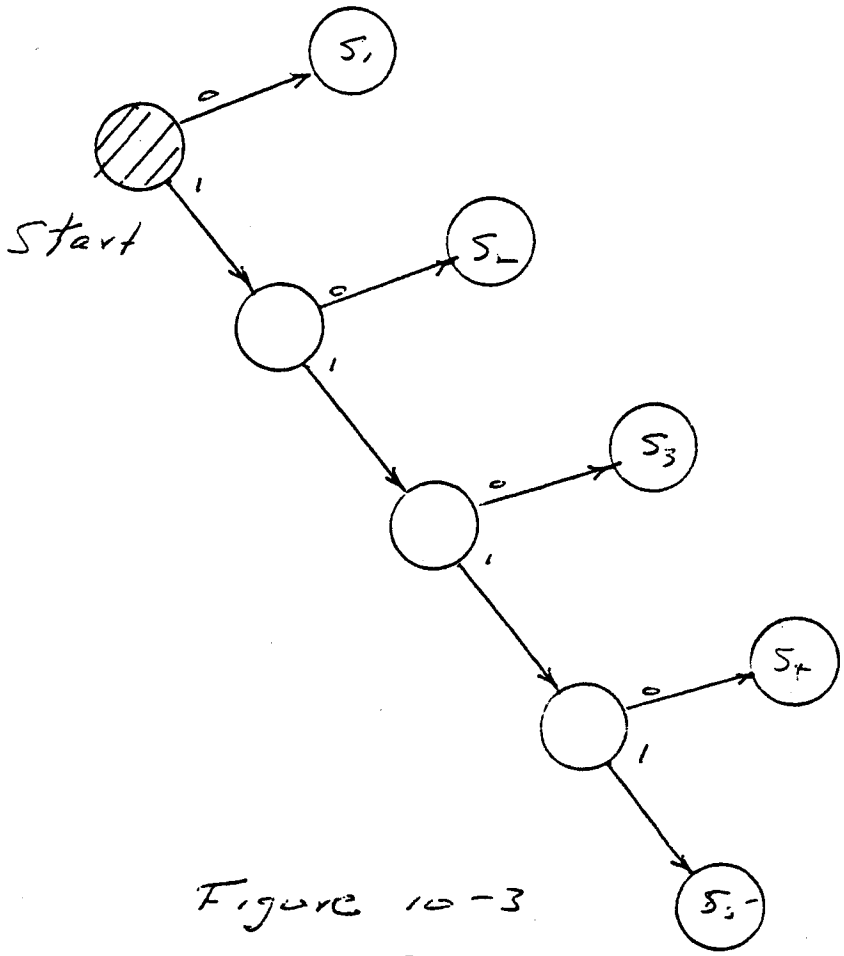


Figure 10-3

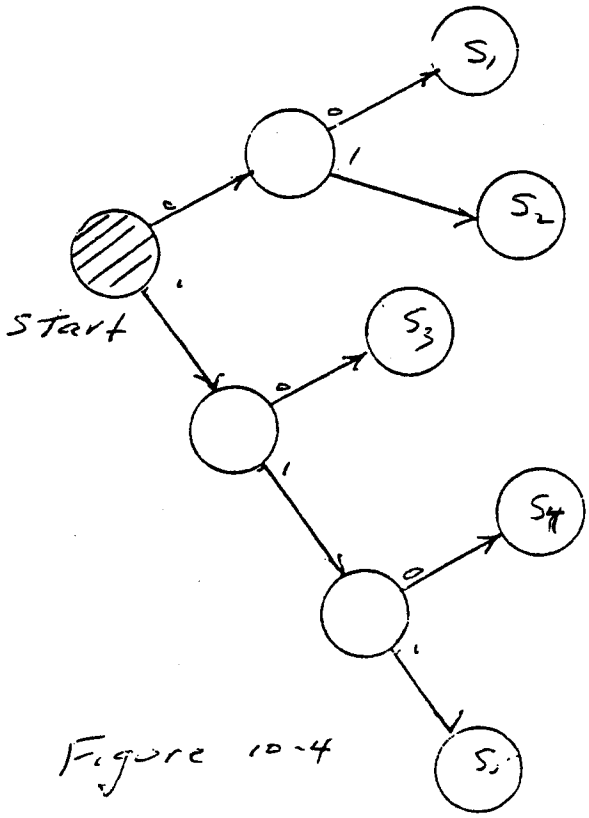
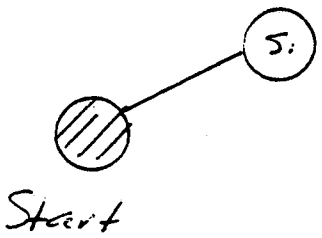


Figure 10-4



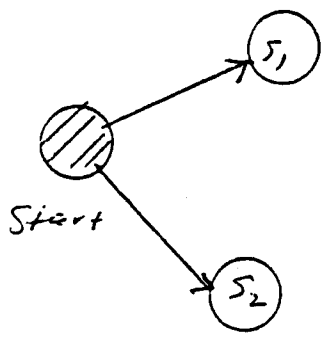


Figure 10-5