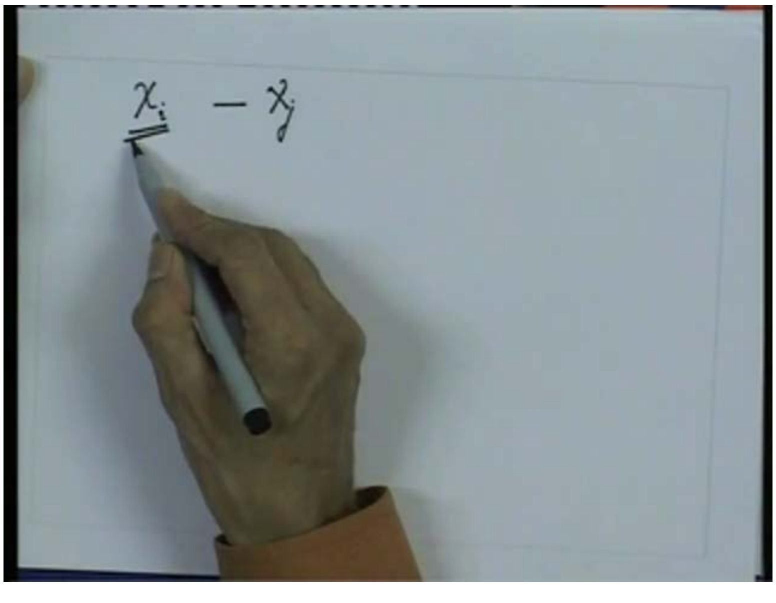
Information Theory and Coding
Prof. S. N. Merchant
Electrical Engineering
Indian Institute of Technology, Bombay

Lecture - 8
Instantaneous Code and Its Properties

Necessary and sufficient condition for a code to be instantaneous is that none of the code word in the cord should be a prefix of another code word in the code. Let us try to prove this, now the sufficient portion of this test is very clear from the definition of the instantaneous code. If you have a codeword, which is not a prefix of another code word, then we can decode the sequence of code symbols in a direct manner. Let us say when we received the code symbol sequence, then we start scanning this sequence until we reach the sub sequence corresponding to a code word.

Now, this sub sequence must correspond to some code word, because we have made the assumption that none of code word is a prefix of another code word in the same code. So, as soon as we receive the subsequence corresponding to a particular code word we can decode it as that subsequence belonging to a particular symbol in the source alphabet, so there is not time lag in the decoding process. To prove the necessary portion of this test, let us assume the contrary, and in order to prove the necessary portion of this test, let us assume a contrary and derive the contradiction.

(Refer Slide Time: 02:46)

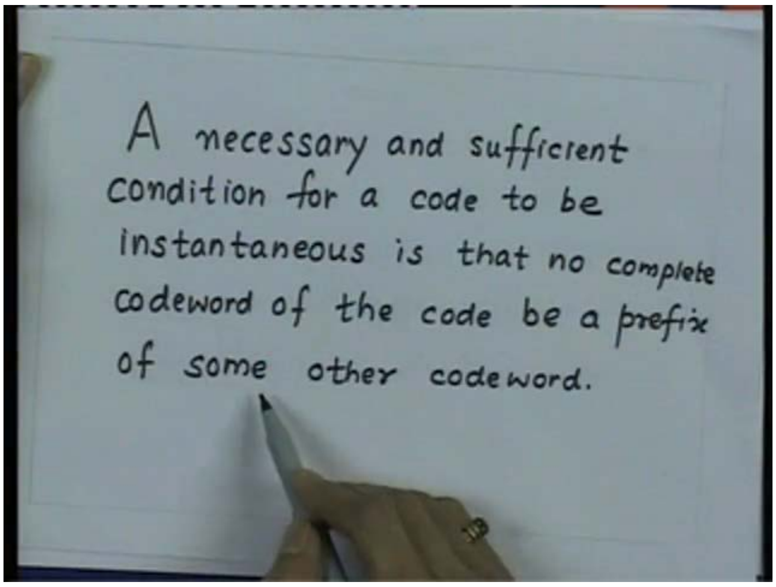


$\underline{\underline{x_i}} - x_j$

So, let us assume that I have a code word x i which is a prefix of some other code word x j in the code. Now, when we received the code symbol sequence, then when we come upon the sub sequence corresponding to the code word x i. Then at that instant we are not sure whether that substituent's belongs to the x i of x j, because we have made the assumption that x i is the prefix of x j.

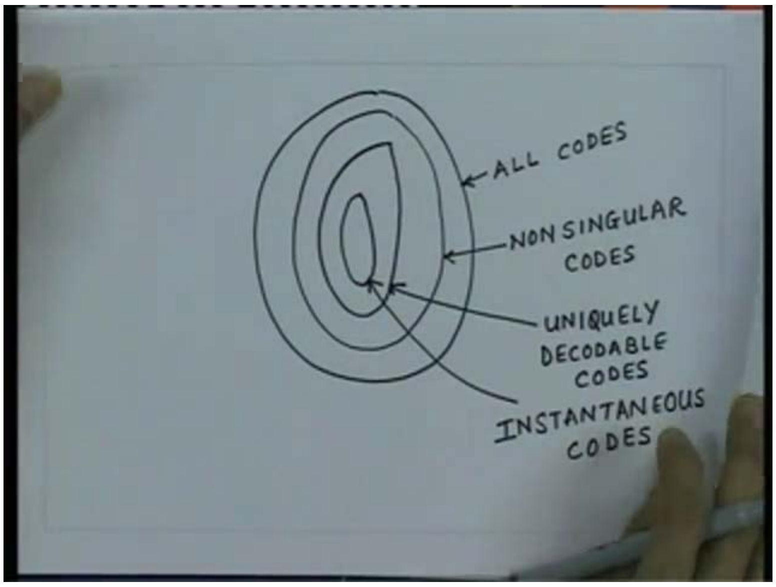
In order to make the decision we have to observe few more code symbols in the main sequence before we take the decision. So, what it means that there is time lag in our decoding process and therefore, the code is not instantaneous. So, we started with the assumption that if x i is a code word prefix of another code word x j, then the code cannot be instantaneous.

(Refer Slide Time: 04:20)


A necessary and sufficient condition for a code to be instantaneous is that no complete codeword of the code be a prefix of some other codeword.

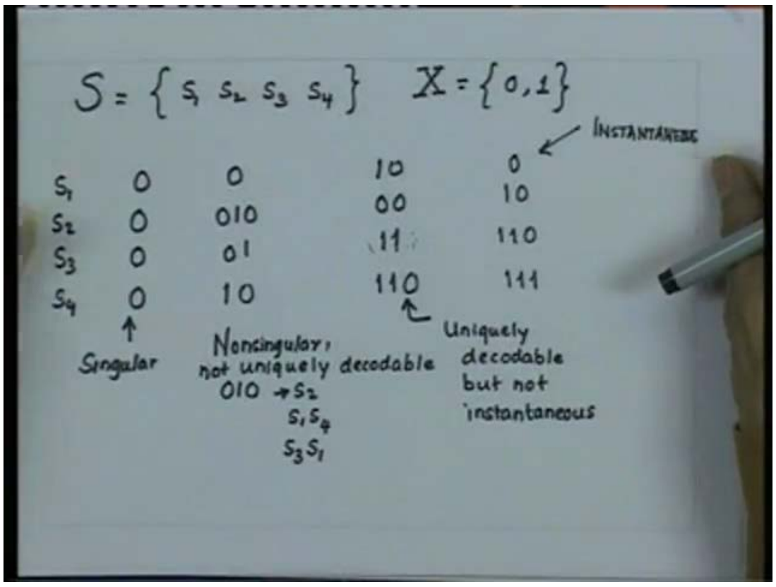
So, this proves both the necessary and sufficient condition of the tests that for a code to be instantaneous no complete code word of the code should be a prefix of some other code word.

(Refer Slide Time: 04:44)



Now, let us summarize what we have talked about codes, in summary we can say that we have a class of all codes and out of all these codes we are interested in a sub class of all these codes and that is non singular codes. We have also seen that just the constrain of non singularity is not sufficient for us to decode the sequence of code symbols. So, what we want that code should be also uniquely decodable and that forms another subclass of non singular codes and this is a subclass and that is uniquely decodable codes. Finally, we said that what we would be interested is, in another subclass of this uniquely decodable code and that is instantaneous codes. So, let us summarize all this with the help of a simple example.

(Refer Slide Time: 06:32)



$S = \{s_1\ s_2\ s_3\ s_4\}$ $X = \{0, 1\}$

INSTANTANEOUS

s_1	0	0	10	0
s_2	0	010	00	10
s_3	0	01	11	110
s_4	0	10	110	111

Singular

Noncingular, not uniquely decodable

$010 \rightarrow s_2$, $s_1 s_4$, $s_3 s_1$

Uniquely decodable but not instantaneous

So, let us assume that I have a source consisting of four source symbols and for this source I have different codes. One code is where the code symbols for this source symbols are 0 0 0, and let us assume once that our code symbols come from the binary code alphabet. So, if this code is a singular code, because for each of this source symbol the code words are not unique. Another one would be 0, 0 1 0, 0 1, 1 0, so this I would say it is a example of singular code, where all the code words are same for all the source symbols.

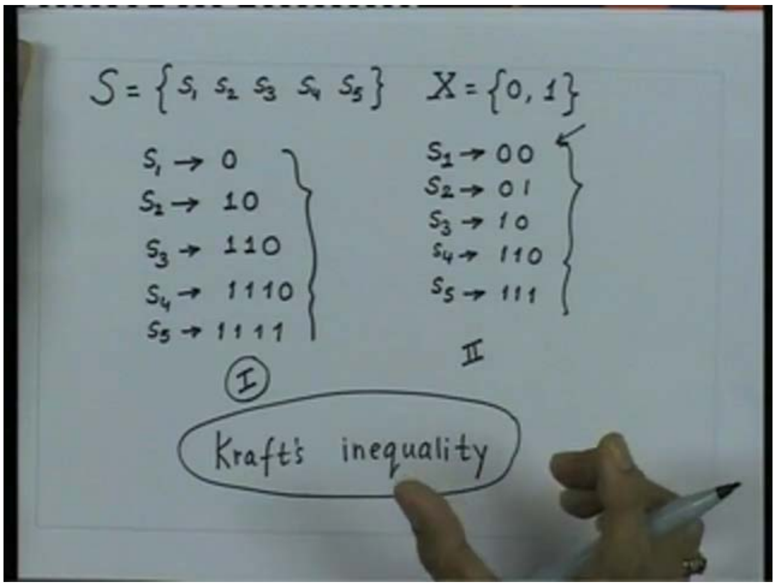
This is an example of a non singular code, but when I received 0 1 0, than I am not able to decode it in unambiguous manner, because 0 1 0 could either corresponds to S 2or it could correspond to S 1 followed by S 4 or it could correspond to S 3 followed by S 1. So, this is another example of a not uniquely decodable code, but this is non singular, not uniquely decodable. Another example would be 1 0, 0 0, 1 1, 1 1 0, this code is uniquely decodable, but this code again is not instantaneous. When I received code symbols sequence, than when I have 1 0 or 0 0, I can uniquely, immediately identify this subsequence 1 0, 0 0, as either S 1 or S 2.

But when I received a subsequence in this sequence of code symbols, where the subsequences 1 1, then immediately I cannot decide, I have to see basically what follows this 1. If 1 follows after this, then I can be sure that this sub sequence corresponds to S 3, but if a 0 follows, then I have to check for the number of zeros which follow after this 1.

If there are even numbers of zeros following after this 1, it means that this subsequence 1 1 must be S 3, but if there are odd numbers of zeros following this 1. Then what it means is that 1 1and the next 0 would be S 4 and other zeros would be decoded as S 2.

So, I have to observe the future symbols when I receive 1 1, so I cannot decode immediately, so this is example of a non instantaneous code. Finally, I have 0 1 0, 1 1 0 and 1 1, this is an example of instantaneous code. In the light of what we discuss earlier we see that none of the code words in this code is a prefix of another code word in the same code. Now, in order to appreciate the nature of constraints on the code, when we want to have this code as instantaneous, let us make some primitive attempts to synthesize an instantaneous code.

(Refer Slide Time: 11:58)



$S = \{s_1\ s_2\ s_3\ s_4\ s_5\}$ $X = \{0, 1\}$

I	II
$s_1 \to 0$	$s_1 \to 00$
$s_2 \to 10$	$s_2 \to 01$
$s_3 \to 110$	$s_3 \to 10$
$s_4 \to 1110$	$s_4 \to 110$
$s_5 \to 1111$	$s_5 \to 111$

Kraft's inequality

So, let us say that I have a source consisting of five symbols and I am interested in designing or synthesizing a binary instantaneous code, so my code alphabet is assumed to be binary and it is 0, 1. So, let us start synthesizing this binary instantaneous code. One way in which we could start is that I can assign 0 symbols to the source symbol S 1. So, my first code word consists of length 1 and it is code symbol 0 that is assigned to S 1. Now, for code word which is to be assigned to S 2, I cannot start with 0, because that will violate the condition for the instantaneous code, so the only alternative left for me is to start with 1.

Now, again I cannot assign a single code symbol 1 to source symbol S 2, because if I did that, then I will be exhausted in the sense that, I will not have any more symbols to start my code words for other source symbol S 3, S 4, S 5. If I want my code words to be instantaneous and for the code word to be instantaneous, none of the code words in the code can be a prefix of another code word. So, I cannot assign 1 to S 2, so the only alternative left for me is to use 1 0. Now, once I have assigned 1 0 to S 2, what is left out is I can use 1 1, now if I have 1 1 again I cannot solely assigned to S 3, because if I did that, than I will not be able to form the code word for S 4 and S 5 in a way where S 1, S 2, S 3, will not be a prefix.

So, one way to assign S 3 would be 1 1 0, now what is left out would be 1 1 1. Now, I have two more symbols, so again I cannot assign this 1 1 1 to S 4, I can assign 1 1 1 and 0 to S 4 and finally, to S 5 I assign 1 1 1. So, this is one way of synthesizing and binary instantaneous code, where I started with a code word length of 1 for S 1 and reached to the code word length of 4 for both S 4 and S 5. Now, from this example it is very obvious that when I start with fewer symbols initially, then for the code words or for the later code words, the length of the code word increases.

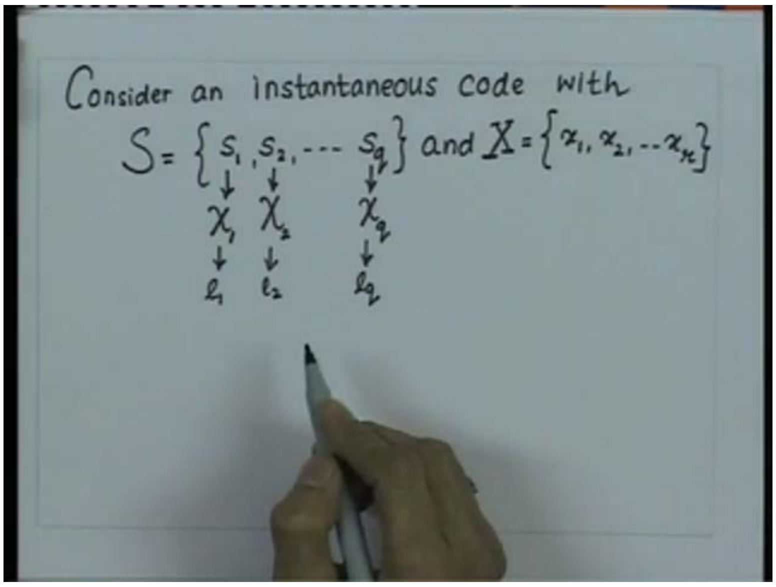
So, what it means that if I were to start with more number of code symbols for my starting source symbol, than I would had more freedom to choose my code words for my later source symbols and probably what would happen that my later code words would not have been so large. So, let us try to test this conjecture with synthesizing another binary instantaneous code where I assign two symbols, code symbols to my first source symbols as 0 0. Now, once I have assigned two code symbols 0 0 to S 1, I have three more possibilities which I can use as an assignment of code words for my remaining source symbols.

Now, since I have four more left and I have three, I cannot assign all the remaining code symbols of length 2 to S 2, S 3, S 4. So, what I will do basically is assign to S 2 as 0 1 to S 3 I will assign 1 0 to S 4 I cannot assign 1 1, because if I did that, if I assign 1 1, than I would not have anything remaining to be assigned to S 5 in a way where none of this code words form a prefix. So, the only way for me to assign S 4 is 1 1 0 and now for S 5 I can assign as 1 1 1. So, this is another code which I generate, another binary instantaneous code which I have generated. Now, if I call this code number 1 and if I say this is code number 2 and if I asked the question, that which of the two code is better, than I cannot answer this question based on the information given to me so far.

So, it is not, but one thing is clear that when I use smaller code word lengths for the assignment of the initial source symbols, than the code word lengths which are left for the assignment to the later source symbols will be larger. So, this is the example out here, but when I start with a larger code word length for the initial symbols, than I get more or less balance code word lengths for all the source symbols. So, in the construction of an instantaneous code, the shorter we make the first few code words of the code, the longer we will have to make the later code words, by using 0 as a code word in the first code word, we restricted all other code words to starting with 1.

Whereas, in the second code when we use 0 0as the first code word, in this case we were able to form later code words by starting with 1 and in addition we could also use code words starting with 0 1. So, this example helps us to understand the constraints which are imposed when we try to design instantaneous codes. Now, what we have seen so far is a qualitatively we have looked at the constraints on the code word lengths of an instantaneous code. Now, the same constraints can be put in a quantitative fashion and that is being done by what is known as Kraft's inequality. So, Kraft's inequality gives us a quantitative constraint on the word lengths of the code words which we can use for an instantaneous code. So, let us look at the Kraft's inequality.

(Refer Slide Time: 21:04)



Consider an instantaneous code with

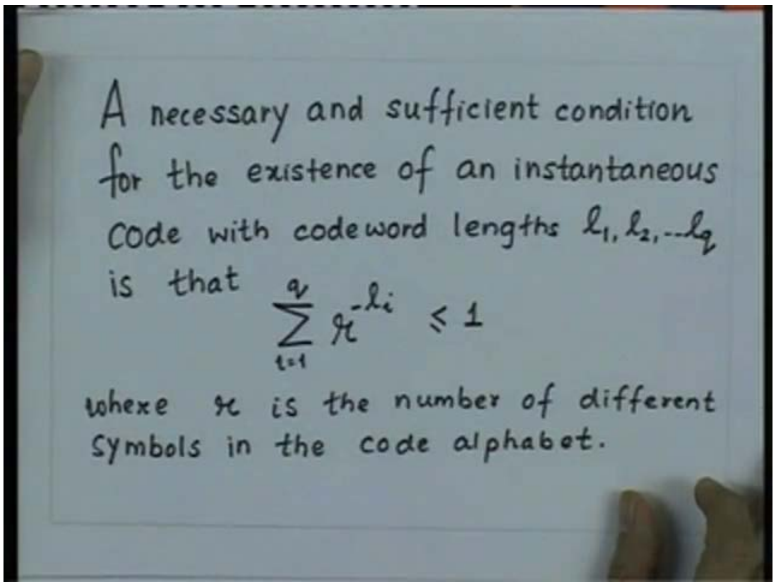
$$S = \{s_1, s_2, \cdots s_q\} \text{ and } X = \{x_1, x_2, \cdots x_r\}$$

$$\begin{tikzcd} s_1 \arrow[d] & s_2 \arrow[d] & s_q \arrow[d] \\ X_1 \arrow[d] & X_2 \arrow[d] & X_q \arrow[d] \\ l_1 & l_2 & l_q \end{tikzcd}$$

So, consider an instantaneous code with source alphabet S equal to S 1, S 2 up to S q and code alphabet this is r symbol code alphabet, so I will have x 1, x 2 up to x r. So, with the help of this code alphabet I form code words for this source symbol. Let me identify this code words as X 1, X 2 and X q, now each of this code words is a sequence of code symbols chosen from code alphabet x. So, each of this will have a length, so l 1, l 2 and l q, l 1 means there are l 1 number of code symbols from this code alphabet which forms a code word x 1 for the source symbol S 1.

Similarly, l q means, there are l q number of code symbols from this code alphabet x, which compromise, which make up the code word x q, for the code for the source symbol S q. Now, it is often desirable that the lengths of the code words of a code be as small as possible. So, now necessary and sufficient condition for the existence of an instantaneous code with code word lengths given by l 1, l 2, l q are provided by the Kraft inequality.

(Refer Slide Time: 23:43)



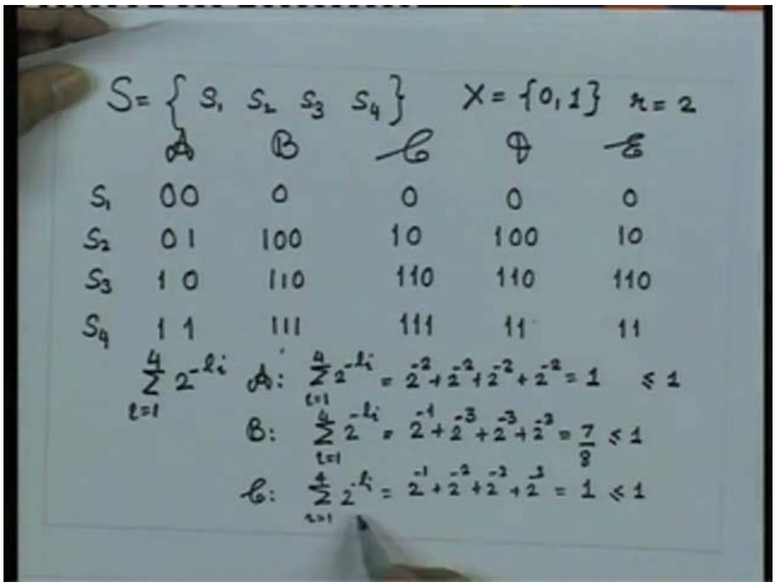
A necessary and sufficient condition for the existence of an instantaneous code with codeword lengths $l_1, l_2, \ldots l_q$ is that

$$\sum_{i=1}^{q} r^{-l_i} \leq 1$$

where r is the number of different symbols in the code alphabet.

So, what is the Kraft inequality says, that necessary and sufficient condition for the existence of an instantaneous code with code word lengths l 1, l 2, l q is that, where r is the number of different symbols in the code alphabet, so this is the Kraft inequality. Now, before we try to prove this Kraft inequality, it will be instructive to look at some examples and see how we might utilize this inequality to decide whether a given sequence of code word lengths can really form the code which is instantaneous. So, with that let us take a simple example.

(Refer Slide Time: 26:12)



$S = \{s_1\ s_2\ s_3\ s_4\}$ $X = \{0, 1\}$ $r = 2$

	A	B	C	D	E
S_1	00	0	0	0	0
S_2	01	100	10	100	10
S_3	10	110	110	110	110
S_4	11	111	111	11	11

$\sum_{i=1}^{4} 2^{-l_i}$

A: $\sum_{i=1}^{4} 2^{-l_i} = 2^{-2} + 2^{-2} + 2^{-2} + 2^{-2} = 1 \leq 1$

B: $\sum_{i=1}^{4} 2^{-l_i} = 2^{-1} + 2^{-3} + 2^{-3} + 2^{-3} = \frac{7}{8} \leq 1$

C: $\sum_{i=1}^{4} 2^{-l_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1 \leq 1$

So, if I have source S consisting of S 1, S 2, S 3, S 4, symbols for this source, let me assume that I have a code which is 0 0, 0 1, 1 0, 1 1, let me say that this code is A. There is another code I have 0, 1 0 0, 1 1 0, 1 1 1, this is my code B. I have another code C which is 0, 1 0, 1 1 0, 1 1 1. I have another code D which is 0, 1 0 0, 1 1 0, 1 1 and finally, I have another code e which is 0, 1 0,1 1 0 and 1 1. Now, if someone gives me this codes, five codes for this source and if the question was to find out whether this codes are instantaneous or not, then Kraft equality by itself cannot help me to decide whether code is instantaneous and not.

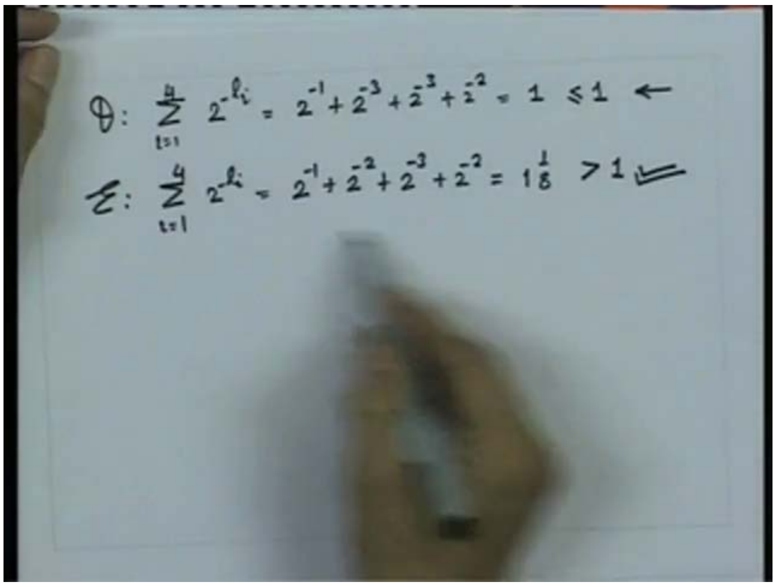
Yes, what the craft inequality helps us to find out, whether the code word lengths which I have chosen are reasonable choice for instantaneous code. Let me take a simple example to explain this, now will try to evaluate the Kraft inequality for each of this five codes, so we are interested in evaluating since our alphabet in this case is binary. So, the value of r we have is equal to 2, so the Kraft inequality will become 2 minus l i, i going from 1 to 4. So, let us try to evaluate this expression for each of this code, so for the code A minus l i, i is equal to 1 2 4 will give me as 2 minus 2 plus 2 minus 2, because each code word is length 2, so what I get is equal to 1.

So, this satisfies my Kraft inequality which says that this quantity should be less than equal to 1. Now, because it satisfies this Kraft's equality, I can be assured that I can have code word lengths of 2 2 2 2, so all the code word lengths can be of equal length 2 for this case. But just because this equality, inequalities satisfy it is not mean that code words are chosen appropriately for the instantaneous codes, for instantaneous code. In this example if you examine the code words for this code A they are also instantaneous. So, it is important to note that the Kraft inequality is the restriction on the word lengths of code and not on the code words itself.

So, this is we get and this code is both instantaneous and it satisfies the Kraft inequality. Let us try to evaluate the same inequality for the code B, for the code B what I would get is if I try to evaluate this 2 minus l i is coming out 2 minus 1 plus 2 minus 3 plus 2 minus 3 plus 2 minus 3 this turns out to be 7 by 8 and again this is less than or equal to 1. So, what it says that code word lenghts of 1 3 3 3 are acceptable as the code word lenghts for instantaneous code. Again if we examine the code words of this code, you will find that they are instantaneous. Example of C is equal to 2 minus 1 plus 2 minus 2 plus 2 minus 3 plus 2 minus 3 and this turns out to be 1 again this is less than equal to 1.

So, again this length, code word length of 1 2 3 3 is acceptable and again if you examine this code words for C, it turns out to be instantaneous, because none of this code words are prefix of other code. Code C was actually obtained from code D by dropping code symbols 0 from the second code word. Finally, let us look at the code D, now this code D is obtained from code B by just dropping a symbol, code symbol 0 from the last code word corresponding to S 4 in the code B.

(Refer Slide Time: 32:10)



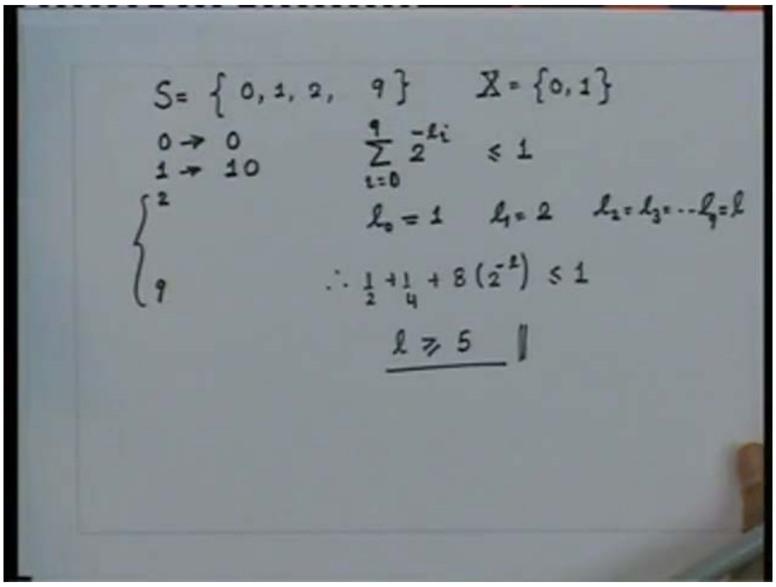
$$D: \sum_{i=1}^{4} 2^{-l_i} = 2^{-1} + 2^{-3} + 2^{-3} + 2^{-2} = 1 \leq 1 \leftarrow$$

$$E: \sum_{i=1}^{4} 2^{-l_i} = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-2} = 1\tfrac{1}{8} > 1$$

Now, for D to evaluate, for D if you evaluate this expression, it turns out to be 2 minus 1 plus 2 minus 3 plus 2 minus 3 plus 2 minus 2 and this turns out to be 1 is less than equal to 1. Now, again is very important to note that what it says that the code word lengths of 1 3 3 2 is acceptable, same out here it was 12 3 3, but if you examine this code D, it is not instantaneous, because code word corresponding to S 4 that is x 4 here is a prefix of a code word x 3, so in no way this can be instantaneous code. But the code word lengths are acceptable as instantaneous code, and finally if you look for the code E and if you evaluate this expression, we get as 2 raise to minus 1 plus 2 raise to minus 2 plus 2 raise to minus 3 plus 2 raise to minus 2 and this turns out to be, now this is greater than 1.

Now, as soon as I have this inequality not satisfied, than I can immediately say that I cannot have this as a instantaneous code, because the code word length itself violates the condition which is required for instantaneous code. Before we look into the proof of the Kraft's inequality, let us consider one more example to appreciate the constraints which go in the synthesizing of a instantaneous code.

(Refer Slide Time: 34:37)



$S = \{0, 1, 2, \quad 9\}$ $\quad X = \{0, 1\}$

$0 \to 0$

$1 \to 10$

$$\sum_{i=0}^{9} 2^{-l_i} \leq 1$$

$$\left\{\begin{matrix} 2 \\ \\ 9 \end{matrix}\right.$$

$l_0 = 1 \quad l_1 = 2 \quad l_2 = l_3 = \dots l_9 = l$

$$\therefore \frac{1}{2} + \frac{1}{4} + 8(2^{-l}) \leq 1$$

$$\underline{l \geq 5}$$

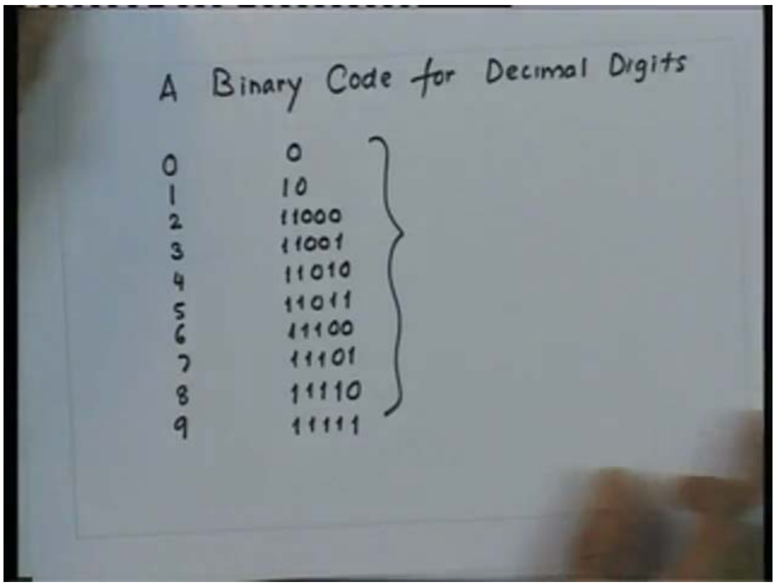
So, let us assume that I have a source consisting of ten symbols and those ten symbols could be decimal digits from 0 to 9 and I am supposed to code this source with a binary alphabet. Let us make one more assumption, that for some reason the occurrence of 0 and 1 is very frequent. So, I want that when I form the code words for the source symbols 0 and 1 the code word lengths corresponding to this code word should be as small as possible. So, let us assume that for my source symbol 0 I assign the code word which is 0 and for my code source symbol 1, I assign the code word as 1 0, obviously I cannot just assign 1 because then I will not able to form the instantaneous code.

Now, for the remaining eight symbols what is desired that I formed the code words which are of equal length. Now, I am interested in finding out what is that minimum length, minimum equal length which I can use for forming the code words for the remaining eight source symbols and I want to form an instantaneous code. So, to answer the question how short it is possible for me to make the remaining eight code words, assuming that all the eight code words are of the same length.

Now, to answer this question we can take the help of Kraft inequality, Kraft inequality says that this condition should be always valid i is equal to 1 to 9 less than equal to 1, since we have made the assumption that the code word corresponding to symbol 0 is of length 1, so l 0 is 1 and l 1 is 2 and l 2 l 3 up to l 9, the code word lengths for this code words are equal, so let us say they are l.

Therefore, I get half plus 1 by 4 plus 8 times l less than equal to 1 and this simplifies to l is greater than or equal to 5. So, what it says that if I want the remaining eight code words to be of equal length and if the code has to be instantaneous, than I cannot form a code which is of length less than 5. So, the minimum length I should have is 5 and it is easy to see that I can form such a code for this source.

(Refer Slide Time: 38:14)

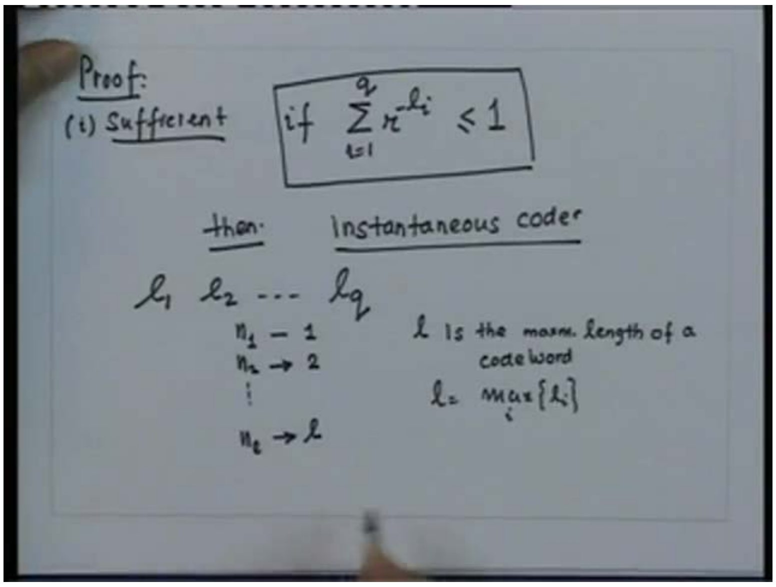


A Binary Code for Decimal Digits

0	0
1	10
2	11000
3	11001
4	11010
5	11011
6	11100
7	11101
8	11110
9	11111

So, a binary code for decimal digits would be. So, this would be the instantaneous code where all the code words corresponding to a source symbol from S 2 to S 9 would be all equal length 5. Now, with this background let us go ahead and try to prove the Kraft's inequality, so the Kraft inequality their two portions, one is a sufficient condition, another is the necessary. So, first let us try to prove so for the sufficient condition, so first is sufficient condition.

(Refer Slide Time: 40:12)



Proof:

(i) Sufficient

$$\boxed{\text{if } \sum_{i=1}^{q} r^{-l_i} \le 1}$$

then: Instantaneous code

$l_1\ l_2 \dots l_q$

$n_1 - 1$

$n_2 \to 2$

$\vdots$

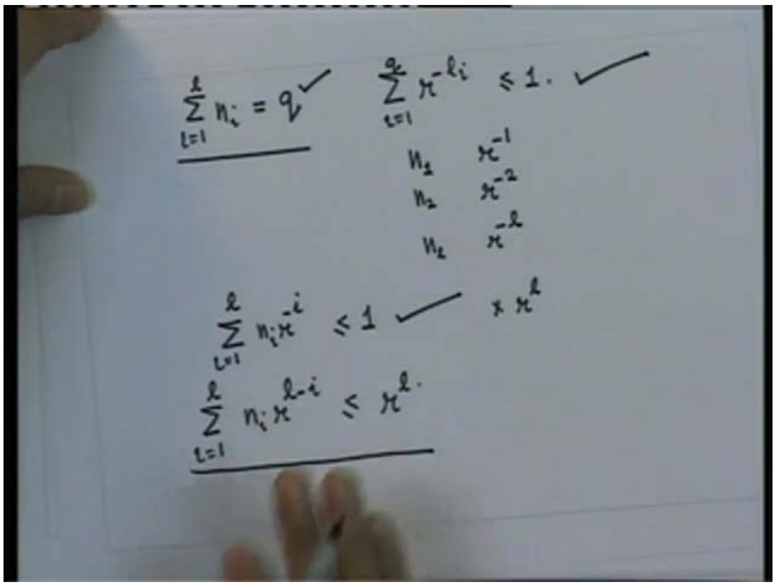
$n_l \to l$

l is the maxm. length of a code word

$l = \max_i \{l_i\}$

So, what sufficient condition says that, if this is satisfied, if this is satisfied then I should be able to construct instantaneous codes. So, let us assume that the required code word lengths are l 1, l 2 up to l q, this code word lengths need not be the distinct. Now, it will be useful for us to consider all the code words of the same length at one time. So, let me assume that I have n 1 code words of length 1, I have n two code word of length 2. Now, let me assume that l is the maximum length of a code word, so l is maximum of l i. So, if I assume this then let me assume that n l corresponds to a number of code words of maximum length l.

(Refer Slide Time: 42:20)



$\sum_{i=1}^{l} n_i = q$ ✓ $\qquad \sum_{i=1}^{q} r^{-l_i} \le 1.$ ✓

$n_1\quad r^{-1}$

$n_2\quad r^{-2}$

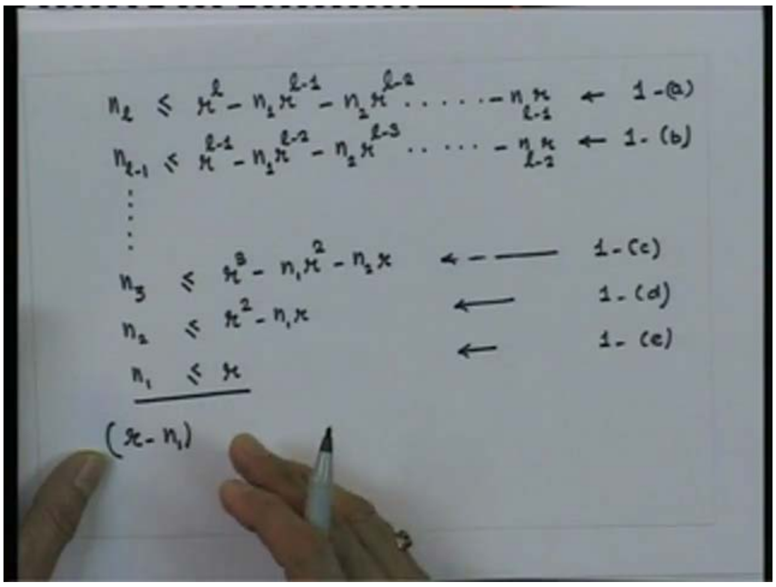
$n_l\quad r^{-l}$

$\sum_{i=1}^{l} n_i r^{-i} \le 1$ ✓ $\quad \times\, r^{l}$

$\sum_{i=1}^{l} n_i r^{l-i} \le r^{l}.$

So, it means that summation of n i, i is equal to1to l is obviously equal to number of code words which I can have, the number of code words will be equal to q and I have this condition. Now, using this equation we can re-write this equation which corresponds to the Kraft's inequality, I assume that this is true. So, what it means that there will be in this inequality based on this relationship here, I will have l 1 terms corresponding to r minus 1 and I will have n 2 terms corresponding to r minus 2 and I will have n l terms corresponding to r minus n. So, based on this I can write this equation out here as, so given this and assuming that the notations we get this relationship. Now, if you multiply this both left hand side and right hand side by r l what I get is, now this forms the basis for a further discussion.

(Refer Slide Time: 44:35)



$$n_l \leq r^l - n_1 r^{l-1} - n_2 r^{l-2} \ldots\ldots - n_{l-1} r \quad \leftarrow 1\text{-}(a)$$

$$n_{l-1} \leq r^{l-1} - n_1 r^{l-2} - n_2 r^{l-3} \ldots\ldots - n_{l-2} r \quad \leftarrow 1\text{-}(b)$$

$$\vdots$$

$$n_3 \leq r^3 - n_1 r^2 - n_2 r \quad \leftarrow 1\text{-}(c)$$

$$n_2 \leq r^2 - n_1 r \quad \leftarrow 1\text{-}(d)$$

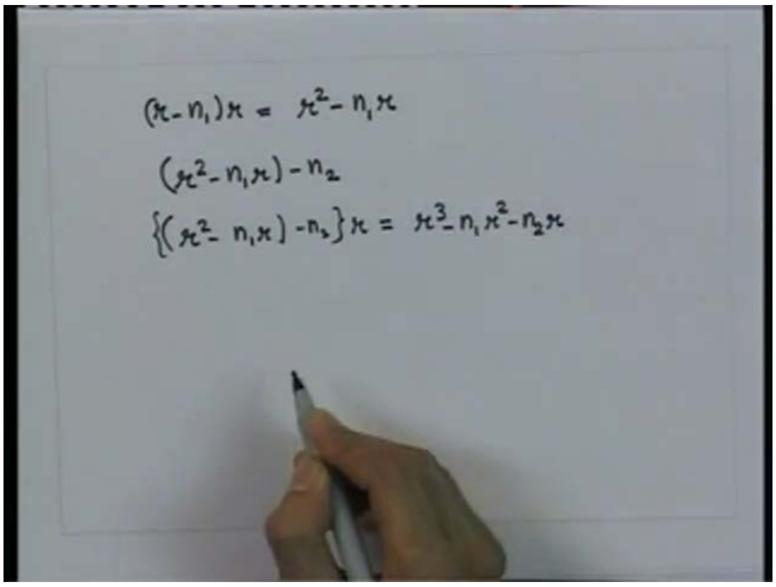
$$n_1 \leq r \quad \leftarrow 1\text{-}(e)$$

$$(r - n_1)$$

Now, let us write out this equation and rearrange terms, so if you write out this expression and rearrange terms we will get n l less than equal to r l minus n 1 r l minus 1 minus n 2 r l minus 2 minus n l minus 1 r, so let me say that this is equation number 1 a. Now, dropping the term on the left hand side and dividing by r I will get as n l minus 1 is less than equal to r l minus 1 minus n 1r l minus 2 minus n 2 r l minus 3 minus n l minus 2 r. So, to drop the term on the left hand side bring this term out here and then divide by r, this is what we will get let us call this equation 1 b. So, this way if you proceed I will get n 3 less than equal to r cube minus l 1 r square minus n 2 r this is 1 c and n 2 less than equal to r square minus n 1 r, and finally I will get n 1 less than equal to r.

Now, this set of inequality is the key to the construction of the code we seek; now what this inequality says that we are required to form n 1 code words of length 1. Now, if we assume that our code alphabet is of size r, then there are r possible such code words that we may form using an r symbol code alphabet and since the requirement is that n 1 should be less than r, we can select this n 1 code symbols of length 1 arbitrary. So, if you do that then we are then left with r minus n 1 permissible prefixes of length 1 namely just those prefixes which were not used as code words. So, now these are the prefixes which we can use, now by adding one symbol to this prefixes we will get code words of length 2.

(Refer Slide Time: 48:24)



$$(r - n_1)r = r^2 - n_1 r$$

$$(r^2 - n_1 r) - n_2$$

$$\{(r^2 - n_1 r) - n_2\}r = r^3 - n_1 r^2 - n_2 r$$

So, the number of code words which we can get of length 2 would be equal to this, r minus n 1 r and that is equal to r square minus n 1 r. So, these are the code words of length two which we can form. Now, if you look at this equation out here, what it says that the number of code words of length 2 required is always less than equal to r square minus n 1 r and we have this available with us.

So, as before we choose our n 2 code words arbitrary from among these choices, so if you choose that the number of prefixes which will be of length 2, which will be left out would be r square minus n 1 r minus n 2. So, this will be the un used prefixes of length 2, now using this unused prefixes of length 2, if we add one more code symbol we can form code words of length 3 and that would be given by r q minus n 1 r square minus n 2 r.

So, this are the permissible prefixes of length 3, again if you look at this equation we are assured that we shall not need no more than this number and we select our code words of length 3 arbitrary from among this. Now, we may proceed in this fashion until we have formed all the code words for a code and all this equations assure us that at each stage we have enough prefixes left.

So, based on this we can always form an instantaneous code, so we have proved that if this condition is satisfied, then it is possible for us to form an instantaneous code, so this is the sufficient part of the Kraft inequality. Now, next time will try to prove the necessary part of the Kraft equality, it is not very difficult we have to just reverse our arguments and will have a look at the little more in depth in the next class.