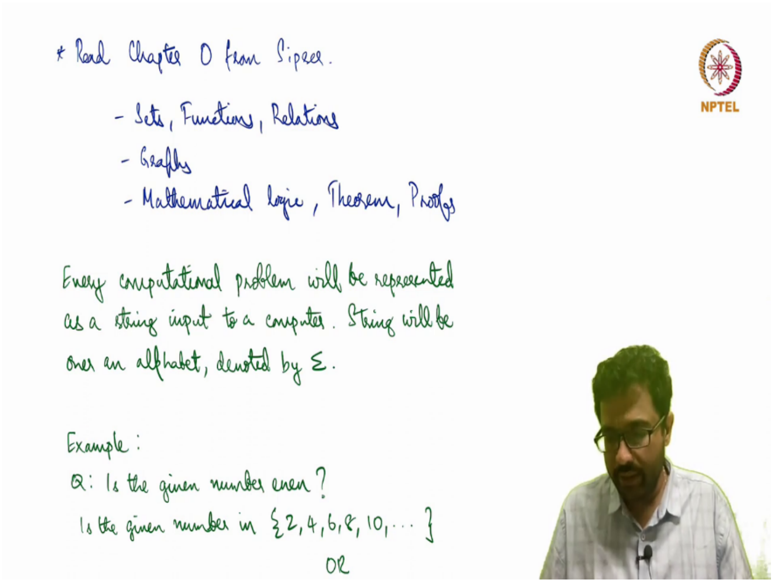
Theory of Computation

Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture 2

Notation and Terminology in Theory of Computation

(Refer Slide Time: 00:16)



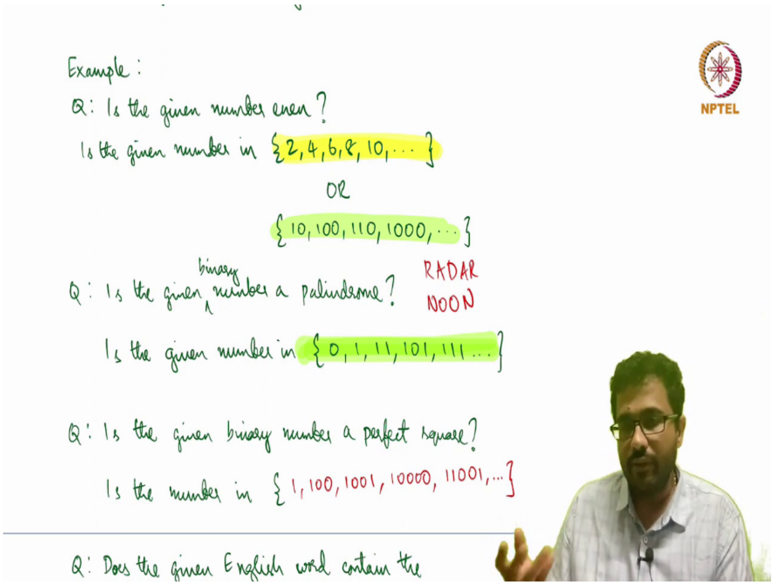
Hello and welcome to lecture 2 of the Theory of Computation Course. So, in lecture 1, we gave a brief overview of the course, and we ended by asking you to read chapter 0 from Sipser, which gives some basics from discrete maths, sets functions, relations, logic theorem proofs, etc. In this lecture, we will actually see some basic information that will help us understand and that will help us set up the notation to understand the rest of the course.

So, the first thing I want to say is that we will usually deal with computational questions, it will be something like, simple computations, things like if you add 25 and 62, what is the sum? Or if you multiply 25 and 62, what is the product? What is the square root of 400? What is the cube root of 400? So these are all computational questions. These kinds of questions also occur in day-to-day life.

So, in our course, we will try to represent all of this as a string input. So in all these questions there is an input, and then we will try to convert, or for most of the course, we will try to view them as a decision problem. So, we will focus on questions of the following type: does this number have a certain property. So for instance one question could be, is the number 400 a perfect square? The answer is yes, because 400 is a square of 20. Is the number 500 a perfect square? And the answer is no, because 500 is not a perfect square, there is no integer which you square and you will get 500.

So, for most of the course, we will consider problems like the above. These are questions of this type: the answer is a yes or no. So, these will be called decision problems because it is a yes or no answer, the answer is not like asking to add two numbers. So the answer is something else, like another number, and so there is an output. We will mostly focus on decision questions.

(Refer Slide Time: 02:42)



So, the decision questions can be represented like this, so for instance, one question could be: is this number even? So, I could write it in the following way, I could give a set like this and ask if the given number is in the following set. So which means if it is a positive integer, is it in {2, 4, 6, 8, 10, …etc}, so it is like the same question in another form. Or if it is a binary integer, then it is like asking if the given integer is in {10, 100, 110, …}. So these are like 2, 4, 6, 8, in binary, etc, so is the given number in this set, is the question that we have.

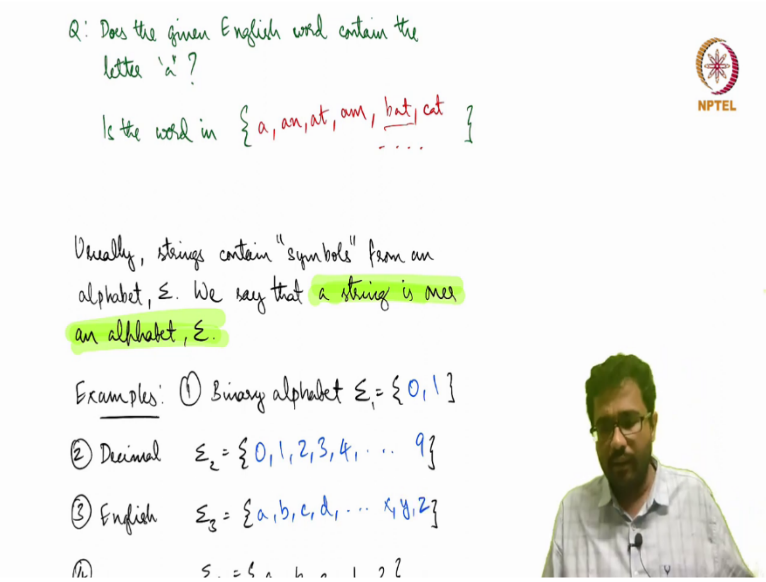
So, the point I want to mention here is that, so you have an input which is a number, and you have a set which is a collection of numbers, and you are asking whether the input belongs to the collection, so this is another way to model the decision problems. So, is this number a perfect square, so you can give a list of all the perfect squares, this will be an infinite list but you can ask if this number belongs to this list.

Another question, is the given binary number, so it is a binary number is given, is it a palindrome? So, palindromes are those numbers or are those strings that read the same from left to right and right to left. So, for instance the word RADAR is a palindrome, R-A-D-A-R reads the same from left to right as well as right to left.

Another word, NOON is a palindrome? N-O-O-N, so if you read it from left to right or right to left, it is the same. So, you can even make palindromic sentences, but let me not get into that. So, the question is whether a given binary number or a binary string is a palindrome? So, another way of asking is a given number in the following set, which is a set of all palindromes.

So, does the given number belong to the following set {0, 1, 11, 101, 111, …etc}? And is a given number a perfect square? So, in other words, it is a given number, let us say the given number is only perfect square, so 1 is 1, the next perfect square is 4, which is like 100 in binary, the next perfect square is 9, which is 1001 in binary, the next is 16 which is 10000, the next is 25, so 25 is 11001. So, I am just giving different examples of questions that we will look at during this course.

(Refer Slide Time: 06:26)



Another question could be, since we have all been dealing with numbers so far, another question could be that does the given English word contain the letter A? So, in other words is the word in the following list where the list is set of all words with the letter A? So {a, an, at, am, bat, cat, and so on…} and so you could fill it up with many many words. This is just some one letter word, and some two letter words, and some three letter words, and so on. But is a given word in this set, so the point I am making here is that all of the problems or most of the problems that we will consider in this course will be of this type, and will not be of the type like given these numbers, you compute something. It will mostly be decision problems.

So, you have an input which is composed of some strings, some number or some word or whatever, and you have a set and you want to see whether the input belongs to this set. So, the first three questions here, can we use the input as well as the sets which were binary meaning the numbers in the sets were binary? In the last question, the input was a word, an English word, and the set also consisted of English words.

So, now why I am saying this is that this will help us set up some basics that we will use throughout the course. So, the binary used zeros and ones, and the English word used English symbols, so these things are called alphabets. In binary, the alphabet is just two digits or two bits, 0 and 1, and the English alphabet is all the 26 English letters, so this will help us set up some basic terminology that we will use throughout the course.

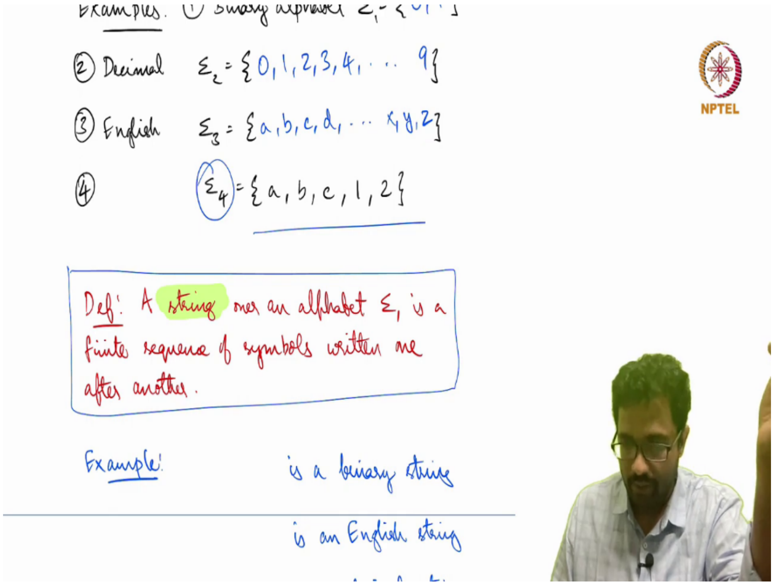
So, we will deal with strings, which are just a bunch of symbols. So these are all like a bunch of numbers like 0, 100 or 01000, 1001 and all of these are strings over their respective alphabets. So, like bat is a string, BAT, it contains three symbols B, followed by A, followed by T. So, strings contain symbols from a certain alphabet, and another way to say the same thing is that a string is over an alphabet. So we say that a string is over an alphabet, meaning a string contains the symbols from a certain alphabet.

So, what are some standard alphabets that we will consider? So one is the binary alphabet which contains 0's and 1's, another one is decimal alphabet which contains all the ten digits, 1, 2, 3, 4, up to 9, then we have the English alphabet which contains all the English letters a, b, c, d, up to x, y, z.

Or you could just have some alphabet defined like this, which contains some three letters a, b, c, and two numbers 1 and 2. So this alphabet Σ_4 contains only five symbols. The binary alphabets contain only two symbols, and so the members of the alphabets are called symbols. So binary alphabets contain two symbols, decimal alphabet contains ten symbols, and the English alphabet contains 26 symbols.

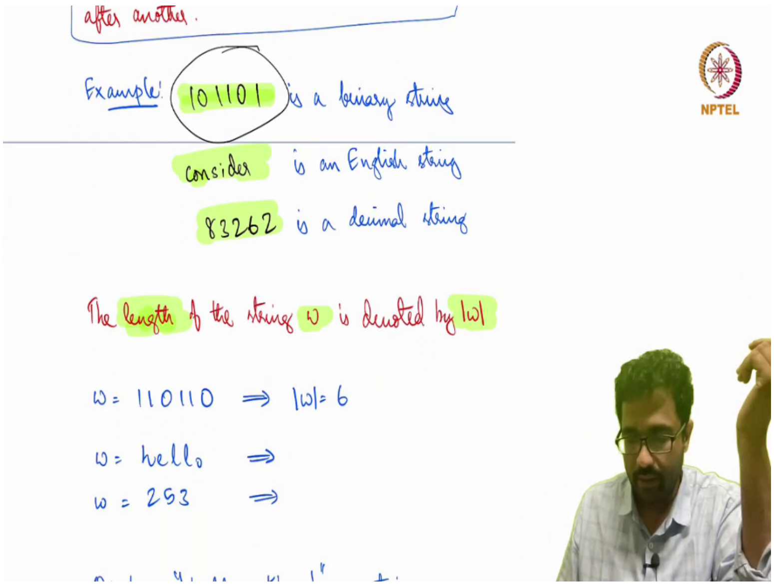
And usually, we denote the Alphabet by using the sigma notation, so Σ_4 contains five symbols, and strings use symbols from these alphabets. So in any binary string we repeatedly use 0's and 1's, in English words we use letters from the English alphabet.

(Refer Slide Time: 11:05)



So, a string over an alphabet, formally defining the same thing over an alphabet, is a sequence of symbols written one after the other. So this is in definition of a string, it is a finite sequence of symbols written one after the other.

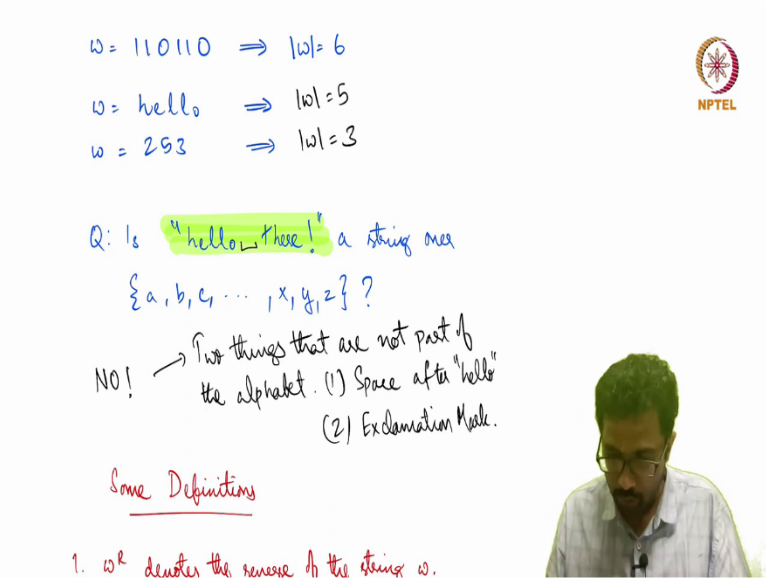
(Refer Slide Time: 11:23)



Just some more examples, so 101101 is a binary string, the word consider is an English string and 83262 is a decimal string. Which means it is a string composed of symbols from the respective alphabet. So, the first one consists of string symbols from the binary alphabet, this has symbols from the English, and this has symbols from the decimal alphabet.

One thing that this small observation is that the first one, in the 101101 is also a string from the decimal alphabet, because it contains the symbols. They are also present in the decimal alphabet. So, given a string, the length of the string is denoted by a W with these two bars around it for W being the string.

(Refer Slide Time: 12:55)

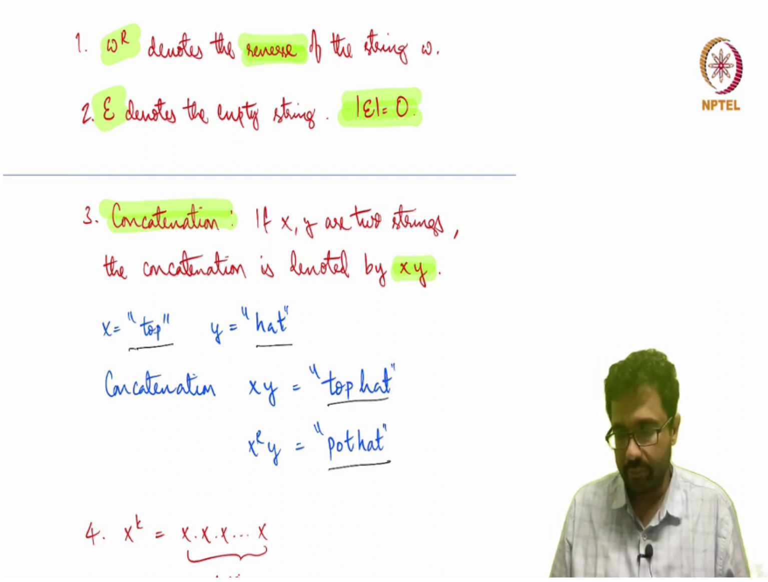


So, for example 110110, the length is 6 because it contains six symbols. The word hello over the English alphabet, the length is five, the string 253 over the decimal alphabet has a length 3. So, let me ask a question, so consider the word or consider the string “hello there!”,. Is this string over the English alphabet, again reminding you, the English alphabet is the set of all the 26 letters a b c d up to x y z. So the question is this string “hello there!”, is a string over the English alphabet?

And the answer is no, because there are two things that are not part of the English alphabet, one is the space after hello, and the exclamation mark. So even the exclamation mark is not part of the alphabet, so this string is not part of the alphabet given here.

But we can make it part of the or so this thing we can consider space and exclamation to be part of an alphabet. So if there is a bigger alphabet with space and exclamation mark as part of that alphabet, then this can be considered as a string which is over that alphabet. So, we can expand the alphabet to make this string over that alphabet.

(Refer Slide Time: 15:52)

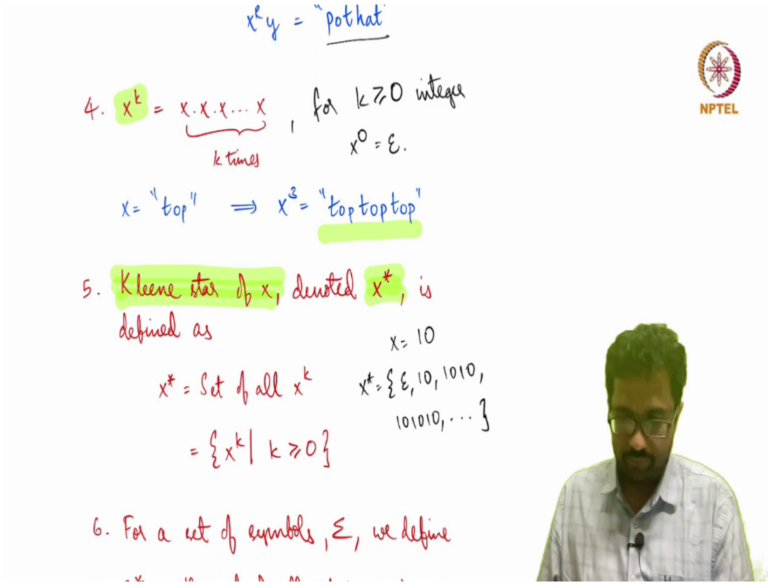


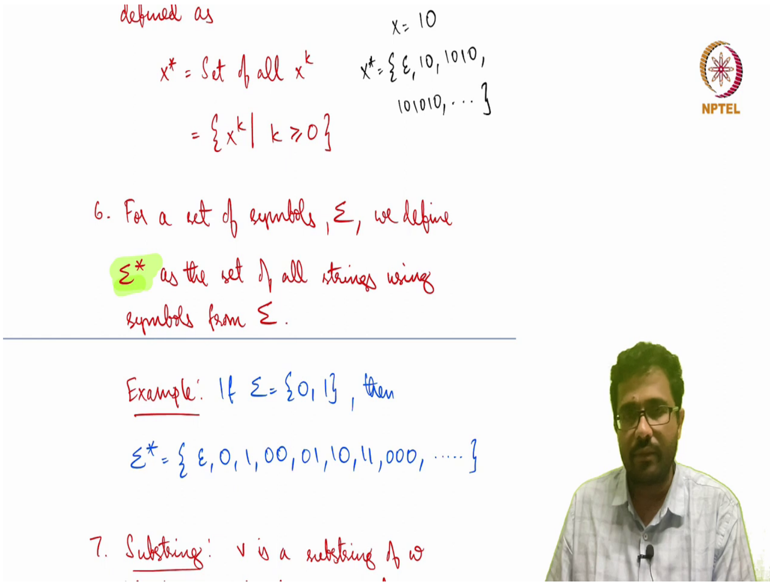
Now, let me get into some definitions which will be useful and we will remind them when some of them need to be used again but still it will be good to go over them once. So, one is w is a string, w^R denotes the reverse of the string, so for instance if W is the word top, then w^R is a reverse which is pot. Another notation is $|\varepsilon|$, this denotes the empty string.

And one convention is that what is the length of the empty string, so any string contains a certain length or has a certain length, the word hello has length five, the word empty has length five, word string has length six, so what is the length of the empty string? The empty string has no letters or no symbols, so an empty string by convention is said to have length 0.

Then some more definitions, one is concatenation. Concatenation is where we append one string after the other. So, if x and y are two strings the concatenation of x followed by y is obtained by merely appending y after x. So, xy denotes the concatenation, so it is just written xy. So if x is about top, and y is the word hat, the concatenation of xy is written just by xy is the string “tophat”. So I hope this does not appear to look like a space, it is just one word, six symbols. Just to give an illustration the concatenation of x^R or the reverse of x followed by y is a string “pothat”, because x^R is pot followed by hat.

(Refer Slide Time: 18:21)

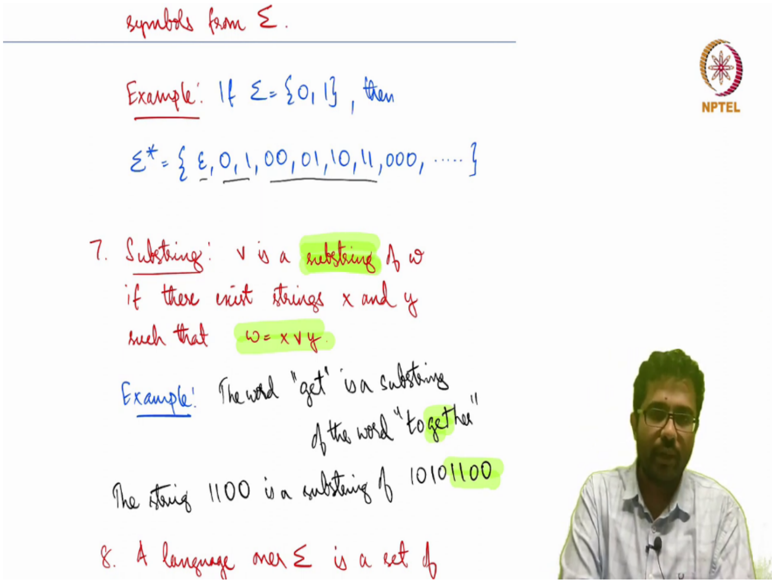




Another notation is the usage of x to the power k, x^k, this is just x repeated k times or x concatenated with itself k times. So if x is the word top, x to the power 3 or x cubed as you may read it for numbers is merely x repeated three times, the word toptoptop. So the x^k is the word x repeated k times. So, k is a given integer, and one more interesting point, so x^0 which means x is repeated 0 times, is actually the empty string but which is not of much use.

Next is Kleene star, so the Kleene star of a string x it is also denoted by x^* is defined as the set of all x^k for all the possible k. So, for instance, if x is 10, then x^* would be the {ε, 10, 1010, 101010, …}. So, this contains $\{x^0, x^1, x^2, x^3, \ldots\}$. This is said to be the Kleene star of a string x. So, these are all important definitions that we will use.

(Refer Slide Time: 20:38)

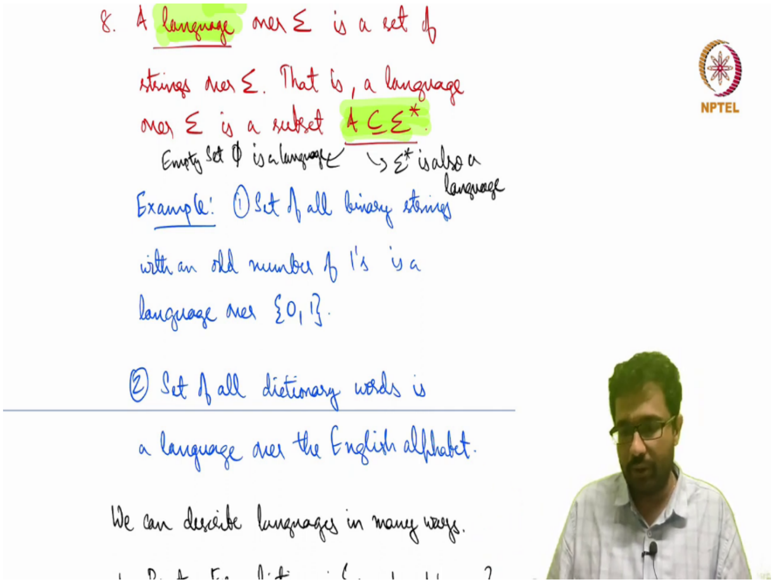


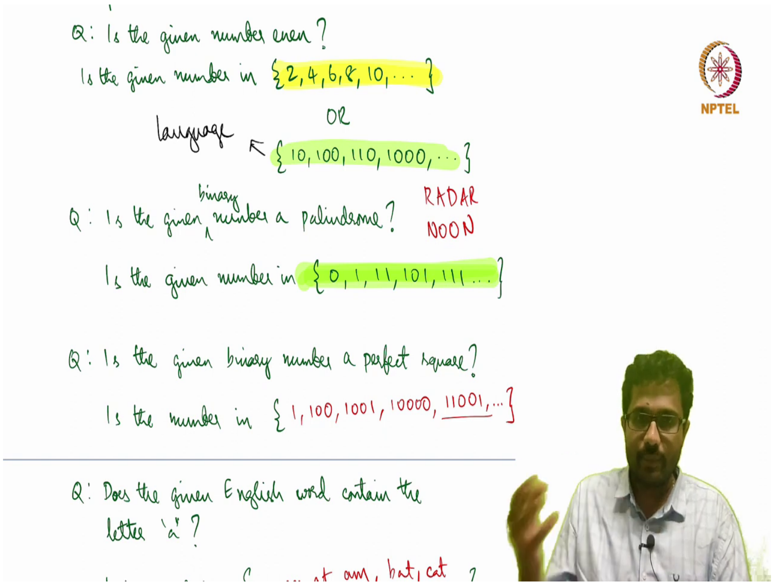
And we even define the star notation the Kleene star for over a set of symbols. So if Σ is a set of symbols or Σ is the alphabet, Σ^* is the set of all strings using symbols from Σ. So, for instance if Σ is the binary alphabet, Σ^* contains the set of all possible strings using 0 and 1. So, you first have the empty string, then you have two strings of length 1, then you have the four strings of length 2, and then you have strings of length 3, and so on.

So, anyway it is a collection and there is no ordering, Σ^* is the collection of all the strings which can be created using symbols from Σ. The definition of a substring V is said to be a substring of another string W, if there exists strings x and y such that W can be written as xvy. Using an example, "get" is a substring of the word "together", because it is "to", and then followed by "get", and then "her". So the word "get" happens to appear in the word "together" like together, so the word "get" appears as it is in the word "together".

And maybe another example is, let us say the string 1100 is a substring of 10101100. So here the substring appears at the very end. So, in this case, the y is an empty string, there is only x, so this is also a substring. So, for any string V appears as part of the string W, then we call V a substring of W.

(Refer Slide Time: 23:35)





The last definition which is also an important one, which is that of a language. So, a language is just a collection of strings over Σ. So it is just a subset of all the possible strings. So, Σ^* as we defined above is the set of all possible strings over Σ, and a language is just a subset of them.

So, for example any subset of them is a language, so set of all binary strings with an odd number of ones is a language over {0, 1}, so it is a set of all binary strings with an odd number of ones is a language because you have all the binary strings and this is a subset of that. So, you have a set of all binary strings which is Σ^* or which is $\{0, 1\}^*$ and this set of all binary strings with an odd number of ones is a subset of them, so that is a language.

Another one is the set, so you have the English alphabet containing the letters {a, b, c, …, z}. These are 26 letters in the alphabet. Now, you can make all kinds of strings using these, so Σ^*

in this case is it of all possible strings. Now, if we focus on the set of dictionary words, the words that are listed in a standard dictionary, this is a subset of them.

Now, the set of all dictionary words is a language over the English alphabet, so if you know the game Scrabble, you end up sometimes, you end up checking whether somebody your opponent lays out some word, and you are not sure if it is an actual word, so then you want to check whether it is there in the dictionary, so you are asking whether the played word is a dictionary word.

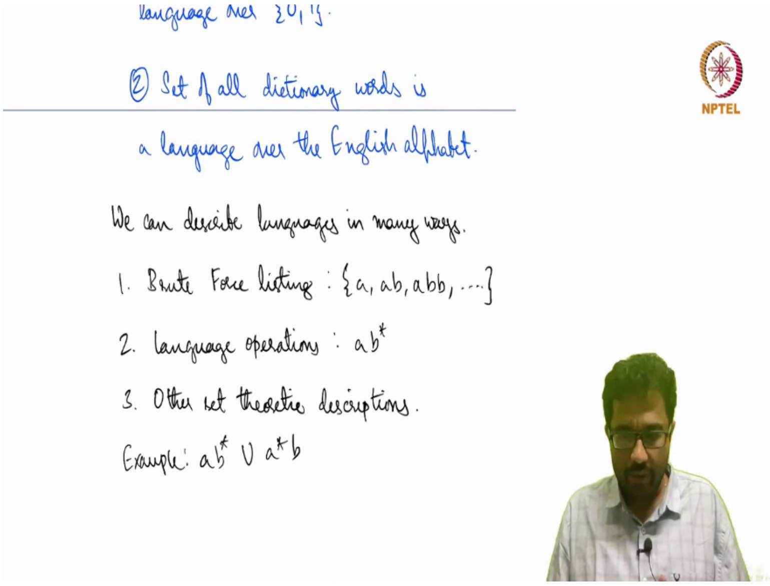
So, we can even consider the Σ^* itself is a language. So this need not be a strict subset or a proper subset. Σ^* is also a language, because Σ^* is itself a subset of Σ^* and it is also a language. The empty set is also a language empty set or sometimes noted like this ϕ is a language. So, the language that contains nothing, and the language that contains everything, these are both the set contains nothing, set contains everything, these are both languages, and everything in between and also languages.

So, the significance of this language is that in the beginning of today's lecture, we said that we will be mostly interested in decision problems, problems of this type: is the given number or the given string does it belong to this collection or does it have a certain property? So, what we have here is that the set, this set over here, lets us say the given number is even, so the first set {10, 100, 110, etc}. So this is a language, so we are asking whether the given string is part of a specific language and the same thing applies for all the remaining problems, for all the remaining questions. So you can consider the language of all the palindromes, and then you are asking whether the given string is part of that language.

So, in our course, we will frequently be referring to the word language, so the language is just a subset of some strings. So, instead of asking whether the given string has a certain property, we will consider the set of all strings that have the property and define a language, define that as a language, and then we will ask if a given string belongs to this language. So, that is why the word language is of importance here, the word language or the notion of language.

So, it is a subset of all the strings, it is a subset of Σ^*. So, giving two examples here but then you could think of various examples. So there are so many examples that the Σ itself is of infinite size. So Σ^* has many possible subsets. Not all of them will have such nice short descriptions in English like this.

(Refer Slide Time: 30:21)



So finally, I just want to say one more thing: we can describe languages in many ways, so one is a Brute Force listing, you could just list down all the members of the language. So, for instance, here I have a followed by ab, followed by abb followed by abbb, and so on, you could just list down the members in Brute Force.

They could have other clever ways of writing like ab^*, denote the same language because b^* contains empty string then b then bb then bbb and so on. So if you append or you add 8 before that you get the strings in the language a, then ab, then abb, and so on. So, this is another way to denote the same language.

You could also use other set theoretic descriptions like you could use operations like this, Union intersection, etc. So, for instance, you could have $ab^* \cup a^*b$, this is the language. So this contains, this is not the same language but then I am using Union to take the unit theoretic Union of two languages, I could even have complement intersection and so on. So, we can use all this to denote languages.

So, I think that is all that I have in lecture 2, where these are some basics that we will, that will help in understanding the remaining content. So, we saw strings, we saw the alphabet, we saw some basics like reverse, empty string, length of the String, concatenation, Kleene star, and so on, then what is the substring, what is the language, etc. And from the next lecture onwards, we will be using these notions kind of regularly, and we will start with some simple models of computation. We will start with what are called finite automata or deterministic finite automata in the next lecture.