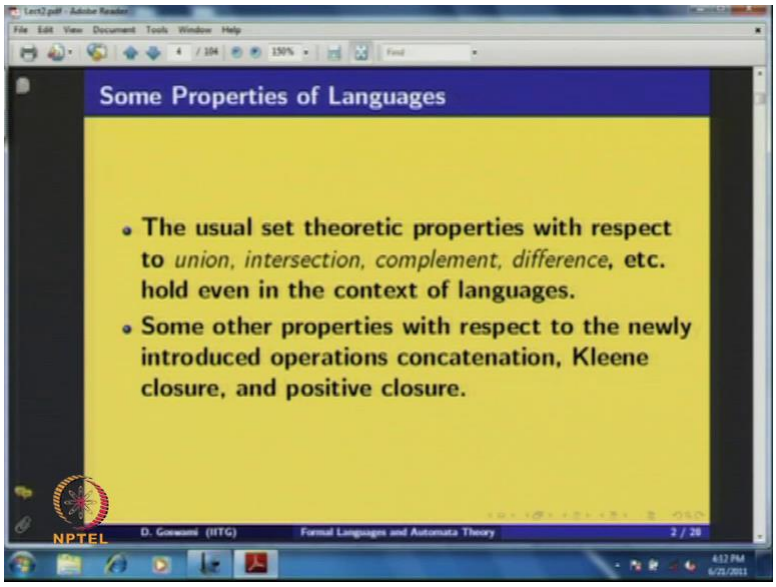
Formal Languages and Automata Theory
Prof. Diganta Goswami
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

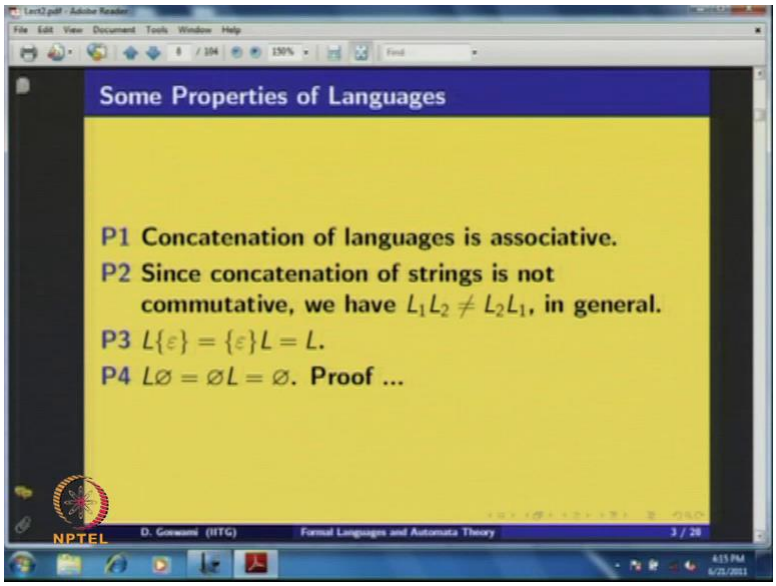
Module - 01
Languages and Finite Representation
Lecture - 03
Finite Representation

(Refer Slide Time: 00:45)



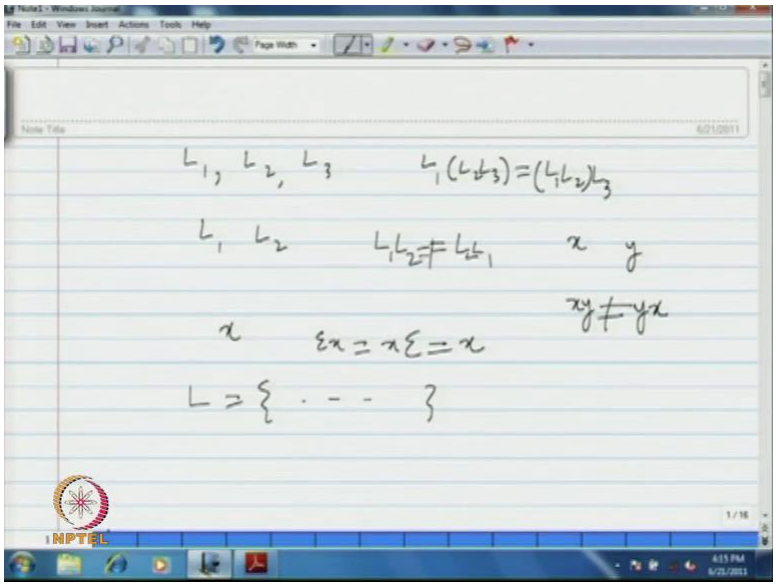
So, for we have defined alphabets, strings, some string operations, and then we have defined formally a language. We have some examples of language also. So, next we will discuss some properties of languages, some the properties we have already discussed. For example, the usual set theoretic properties with respect to union, intersection, complement, difference etcetera hold even in the context of languages. So, we are interested in some other properties with respect to the newly introduced operations concatenation, Kleene closure and positive closure.

(Refer Slide Time: 01:05)



Find that concatenation of languages associative, we have already shown it.

(Refer Slide Time: 01:17)

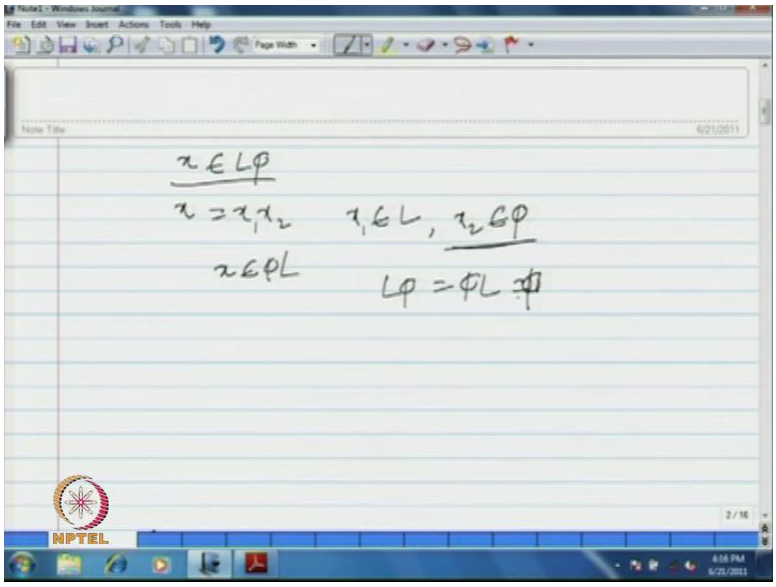


That means, if L 1, L 2 and L 3 are languages. Then L 1 concatenation L 2 L 3 is equivalent to L 1 concatenation L 2 and concatenation L 3, we have already solve it. Now, you know that concatenation of strings is not commutative in general, that why concatenation of languages is also not commutative, that means for 2 languages L 1 and L 2, we cannot say that L 1 concatenation L 2 is equal to L 2 concatenation L 1 there are not identical. Because, in general you know that if x and y are 2 strings, then x

concatenation of y is not identical to y concatenation x turn are identical. And hence L 1 concatenation L 2 is not equivalent to L 1 L 2 concatenation L 1.

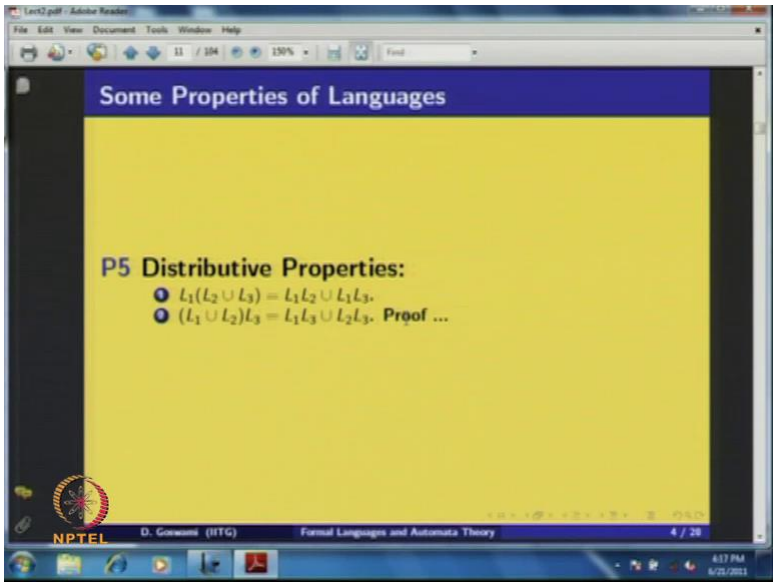
Then is readily to see that, if L is any language then if is concatenate to it a language containing singleton epsilon from both sides left side and right side then it is equal to L, because every string on concatenate to it epsilon from left to right give the string itself, that is if x is string then epsilon x is equal to x epsilon is equal to x. Therefore, every string if L the language every string the language whatever it may be. So, if you concatenate which epsilon from both sides we give the string itself. Therefore, the languages are not going to be changed. Hence we get L epsilon equal to epsilon L equal to L. Then if we concatenate a language with the empty set phi from both sides, L phi or phi L is going to be phi.

(Refer Slide Time: 04:00)



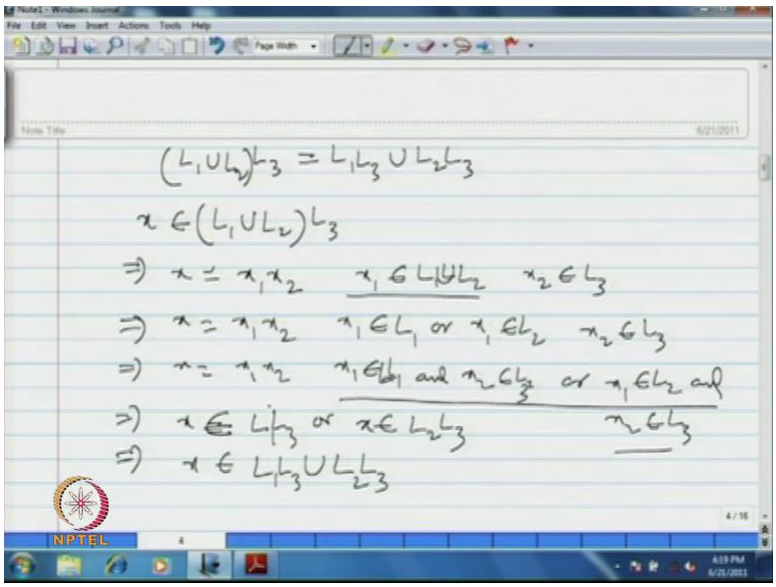
So, we can prove it, suppose that L phi is not equal to phi, then the ((Refer Time: 03:59)) string which will belong to L phi. Therefore, we can write x as x 1, x 2 for some strings x 1 belong to L, and x 2 belong to phi this is the definition of the concatenation, but phi in the empty set that have any string that belong to phi. So, therefore original assumption that x belong to L phi must be long, so L phi must be empty, similarly we can show that x also thus 1 belong to ((Refer Time: 04:45)) any string axe belong to phi, hence the result L phi equal to phi L equal to phi ((Refer Time: 04:55)) of the set.

(Refer Slide Time: 05:01)



Hence see some distributive properties for example, we can show that L 1 concatenation with L 2 union L 3 is equal to L 1 L 2 union L 1 L 2; that means concatenation distributes over union. Similarly, L 1 union L 2 concatenation L 3 is equal to L 1 L 3 union L 2 L 3. Let us gives a proof of the second 1 proof of the first 1 is exactly similar to the second 1.

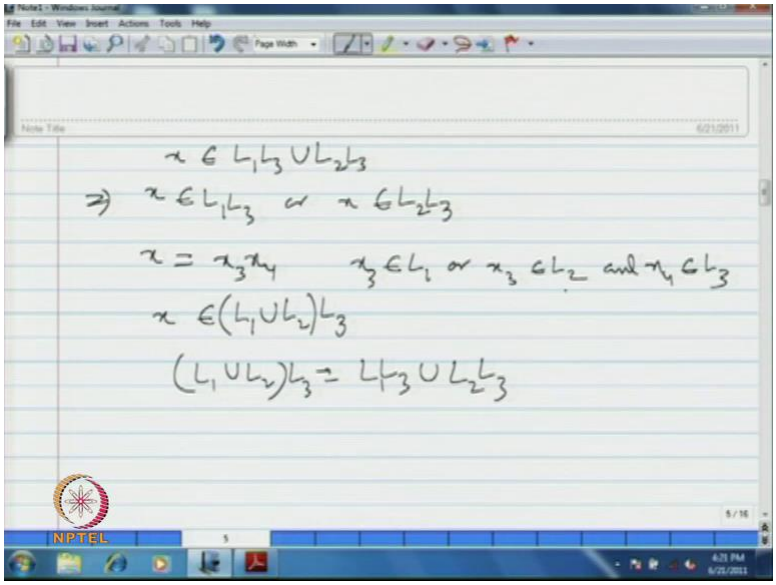
(Refer Slide Time: 05:51)



Suppose, so what in where we show is that L 1 union L 2 concatenation L 3 equal to L 1 L 3 union L 2 L 3. Now, suppose string x belong to L 1 union L 2 L 3, so subsequences

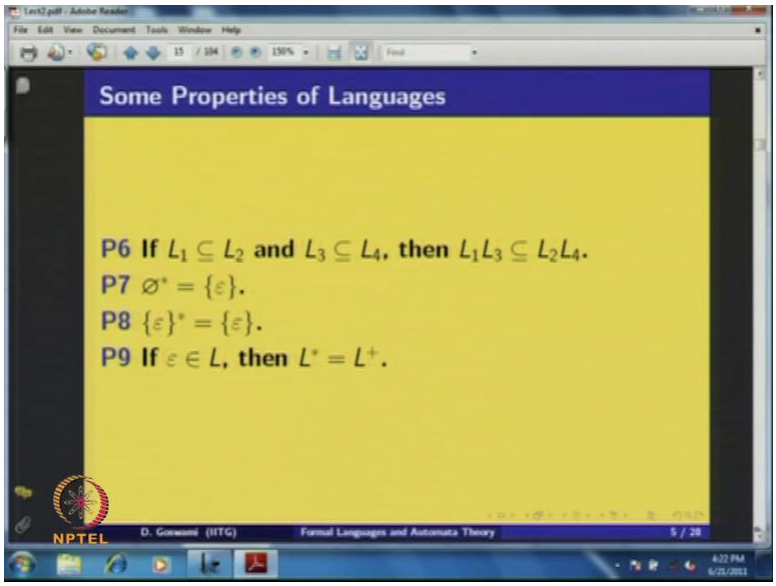
this implies that we can write x as x 1, x 2 for some x 1 belong to L 1 L 2, L 1 union L 2, And some x 2 belong to L 3 this implies x equal to x 1, x 2 for some x 1 belong to L 1 or x 1 belong to L 2, since x 1 belong to L 1 or L 2, and for some x 2 belong to L 3. So, this implies x equal to x 1, x 2 for some x 1 belong to L 1, and x 2 belong to L 3 or x 1 belong to L 2 and x 2 belong to L 3. Now, from this according the definition I can say that x equal to or x belong to L 1 L 3 or x belongs to L 2 L 3. So, this implies that x belongs to L 1 L 3 union L 2 L 3, so this 1 is quite clear. Now, let see the other part the converse part.

(Refer Slide Time: 08:18)



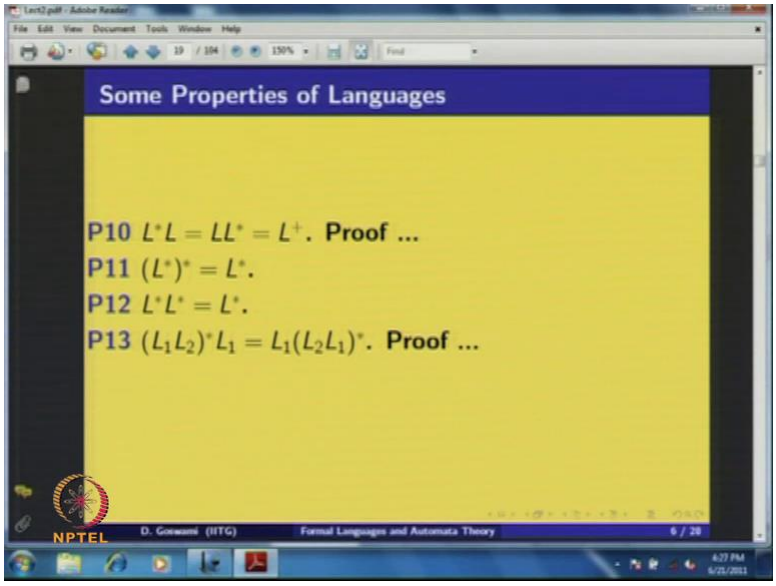
Suppose x belongs to L 1 L 3 union L 2 L 3, now this implies x belongs to, so this implies x belongs to L 1 L 3 or x belong to L 2 L 3, that means we can write x as some x 3 x 4 saws that x 3 belong to L 1 ((Refer Time: 09:04)) or x 3 belong to L 2, and x 4 belongs to L 3. So, according to definition ((Refer Time: 09:17)) this. Therefore, we can write that x belongs to since x 2 belong L 1or x 2 belong L 2 and x 4 belong to L 3. Therefore, we can say that x belongs to L 1 union L 2 and concatenation with L 3, hence L 1 union L 2 from this 2 you can say that L 1 L 2 L 3 is equal to L 1 L 3 union L 2 L 3, so hence the proof.

(Refer Slide Time: 10:09)



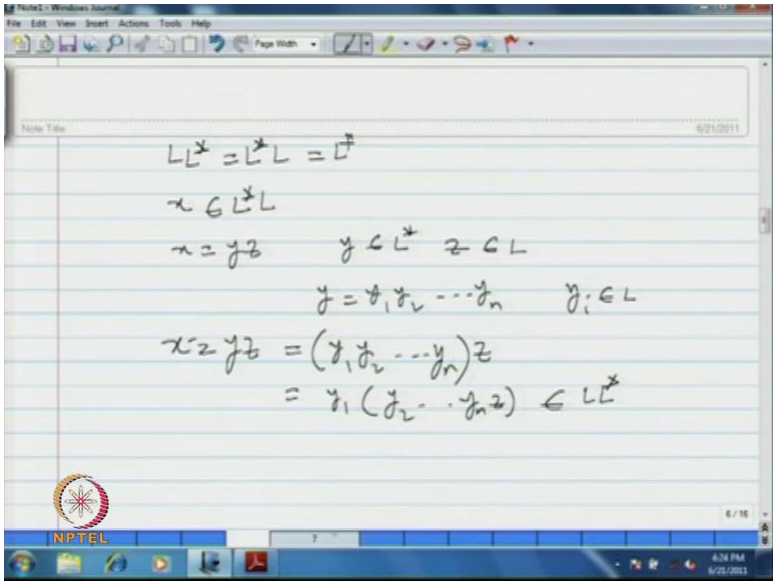
Here see some properties, so all this property number, so the 6 property in the sequence. So, if L 1 subset of L 2 and L 3 is subset of L 4, then L 1 concatenation L 3 is subsets of L 2 L 4 is it proof. Similarly, the other 2 region have show which it prove, so property 7 is phi star is singleton epsilon, so Kleene closure of singleton epsilon gives equal to singleton epsilon, and if epsilon belongs to L, then L star equal to L plus in the positive closure if the string closer equal to positive closer if epsilon belong to L.

(Refer Slide Time: 11:01)



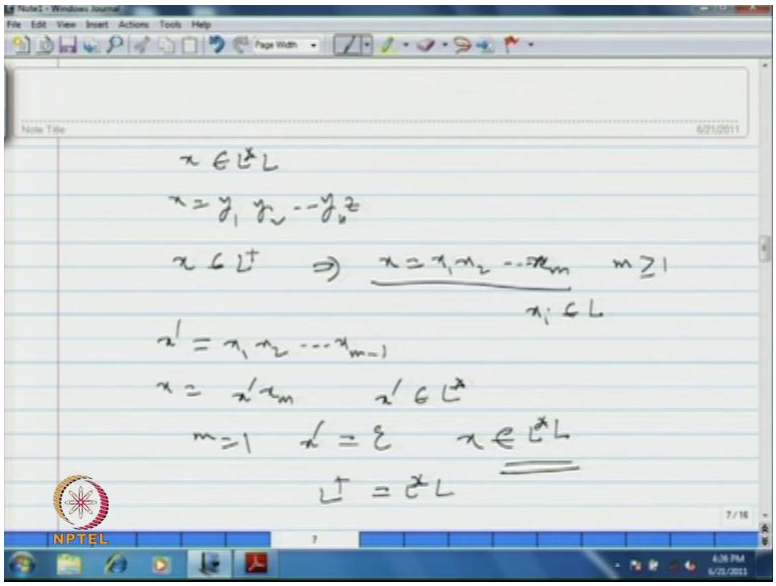
So, next property this L star L equal to L L star equal to L plus you will just provide a proof of this.

(Refer Slide Time: 11:11)



We are went to give a proof of L L star equal to L star L equal to L star and the plus. Suppose x belong to L star L, then we can write x as some y z you can write that x equal to y z for some y belong to L star and z belong to L, but since y belongs to L star which implies that this y can be written as y 1, y 2 and so on y n which y i belong to L. So, this for all i we can write it y as y 1, y 2 up to y n for some y i belong to L. Therefore, x can written as y z this is equal to y 1, y 2 up to y n concatenation over it z is nothing but y 1 then y 2 concatenation with y n and z, this belong to L L star according to the definition. The converse is exactly similar that means L star L equal to L L star.

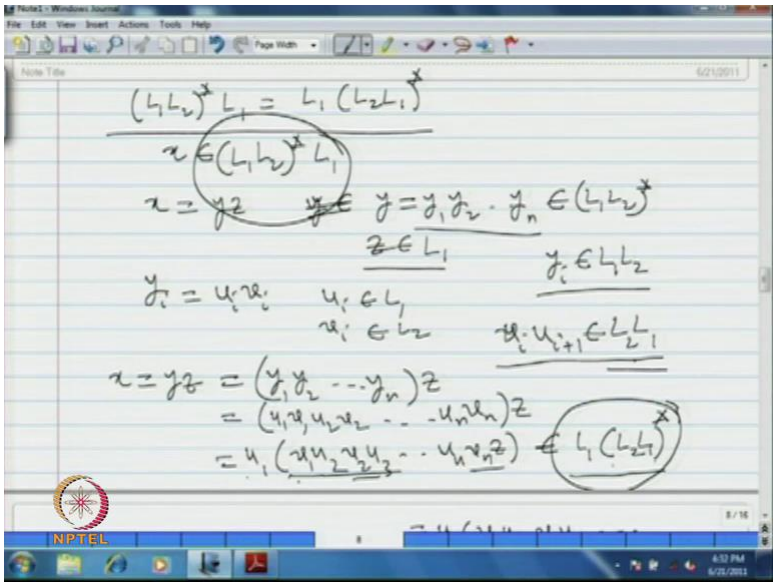
(Refer Slide Time: 13:00)



Further when x belong to L star L it is about, we have x equal to y 1, y 2, up to y 1 z. It is clearly in L plus, so this belong to in L plus. On the other hand x belong to L plus implies that x equal to x 1, x 2, up to m up to x m with m greater than equal to 1, so this are the finishing of positive closure, and here every x i belongs to L for all i. Now, let us write x dash has x 1, x 2, up to z x m minus 1, so that you can write x has x dash x m through this we can ((Refer Time: 13:59)) x dash m. Here note that x star belongs to L star by definition particularly when m equal to 1, if m equal to 1 then x dash equal to epsilon, because m equal to 1 it goes from there is no string, so it is the epsilon.

Thus x belong to L star L. Therefore, L plus is equal to L star L. So, this is the proof for the property. Now, let see L star whole star equal to L star, so we can use the similar concepts to prove this properties and some other similar properties. For example, L star L star equal to L star L 1 L 2 star concatenation which L 1 that is equal to L 1 concatenation which L 2 L 1 star, let us give a proof for this property.

(Refer Slide Time: 15:19)



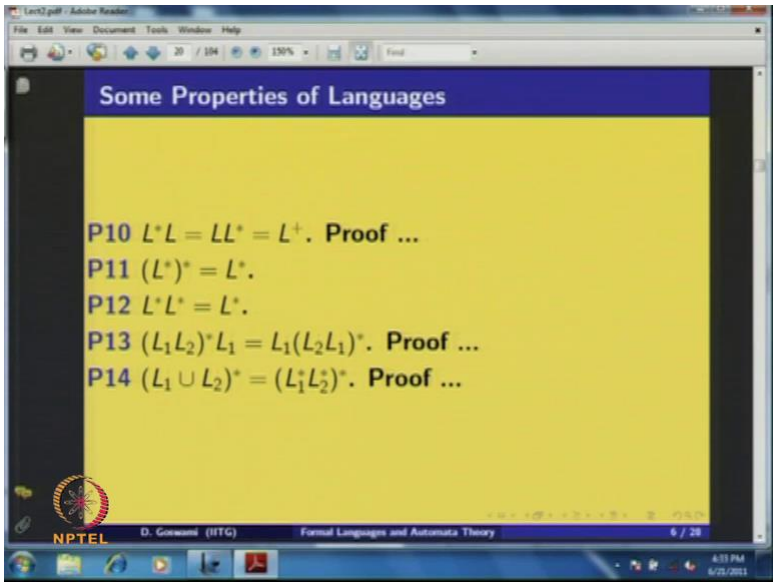
So, we want to prove that L 1 L 2 whole star L 1 is equal to L 1 L 2 L 1 whole star. First let a string x belong to L 1 L 2 star L 1, then we can write x as concatenation of 2 strings y z where y belong to or consider y can be written as y 1 y 2 up to z some y n this belong to L 1 L 2 star, and z belong to L 1. Now, in this case y equal to y 1, y 2, y n it is belong to L 1 L 2 star every y i belong to L 1 L 2 is a concatenation of a string from L 1 and end of string from L 2. Now, each y i and you said can be written a from u i some string from L 1 and v i some string for v 2, some string from L 2 that means u i belongs to L 1, and v i belongs to L 2.

Now, if you can note that the string vi u i plus 1 belongs to L 2 L 1, because quietly under u belongs to u i belongs to L 1, and v i belongs to L 2, therefore v i and u i plus 1must belong to L 2 L 1. Now, we can write x has y z which is equal to y 1, y 2 up to y n z which you nothing but u 1 v 1, because y 1 can be written as u 1 v 1, and y 2 can be written as u 2 v 2 and so on up to u n v n then z. Now, this can be written as u 1 since ((Refer Time: 18:24)) concatenation epoch assertive medulla concatenate of strings, so u 1 v 1 u 2 v 2 u 3 and so on up to v n u 1 v 1 u 2 v 2 u 3 u n v n z.

Now, v 1 u 2 v 2 u 3 this all belongs to L 2 L 1 eventually that v n z belongs to L 2 L 1, since z belong to L 1 and v n belongs to L 2. So, therefore again u 1 belongs to L 1, so therefore this belongs to L 1 concatenation L 2 L 1 star, because they are concatenation from for strings from concatenates strings from L 2 L 1. Therefore, we starting from x

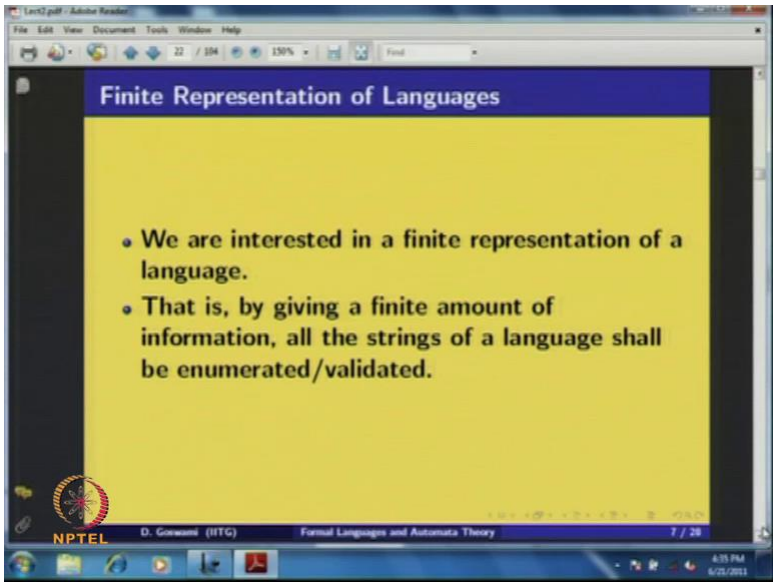
belongs to L 1 L 2 star L 1 a found at x belongs to L 1 L 2 L 1 star. Similarly, you can so ((Refer Time: 19:55)) that means if x belong to L 1 concatenation L 2 L 1 star, it will belong to L 1 L 2 star concatenation L 1. Therefore, the property L 1 L 2 star concatenate L 1 is equivalent to L 1 L 2 L 1 whole star wholes good.

(Refer Slide Time: 20:22)



Let us see some more properties, so L 1 union L 2 star equal to L 1 star L 2 star whole star this proof for these can be taken as an exercise. So, so far a provident many properties up to 14 properties of languages, similarly one can produce many other properties reign the various operations.

(Refer Slide Time: 21:07)

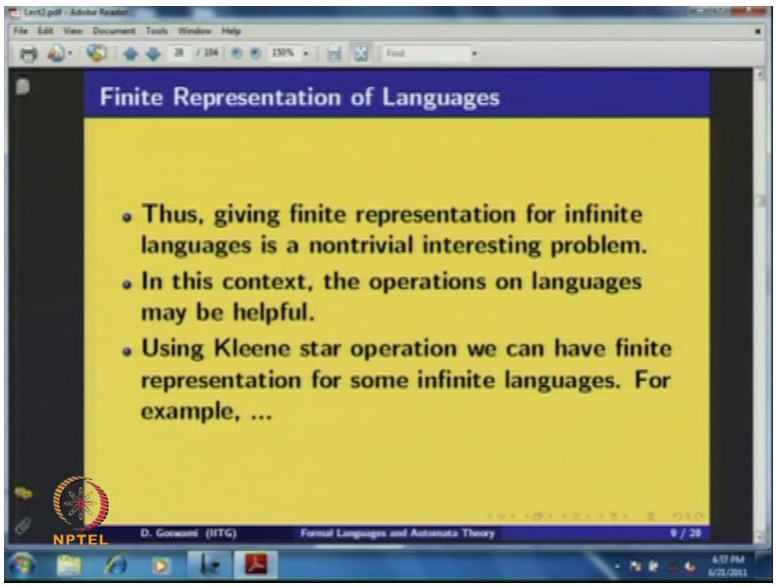


Now, we will see how languages can be represented using infinite information that means finite representation of the languages. A many interested in a finite representation of a language, if a person is proficient in a particular language it does not mean that it produce all the sentences of the languages. Basically what we expect is that using a finite amount of information, we want to be able to valid at or construct difference things being the languages. That means by giving a finite amount of information all the strings of the languages shall be enumerated or validated. For example, if you see the case of compiler, the compiler can validate any program which is nothing but a string from the programming languages using only a finitely valid instructions some ((Refer Time: 22:00)) in it.

(Refer Slide Time: 22:07)

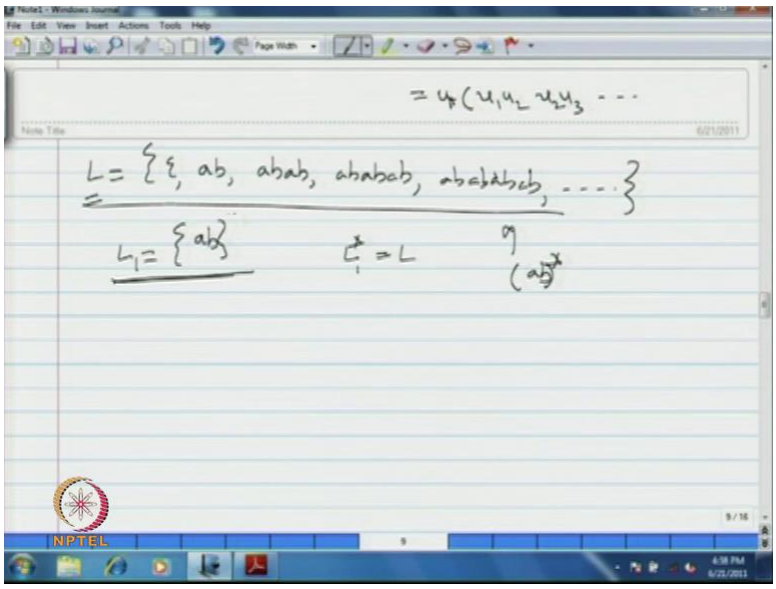
So, there is having we look at the languages for which finite representation is possible. Given an alphabet sigma, the languages with single string x and phi can have finite representation. For example, suppose for a language containing a single string x say x is the finite representation, and for the empty set. So, empty set itself is a finite representation. And any finite languages can also be given a finite representation simply by enumerating all the strings in it. So, do that a finite representation for the different languages.

(Refer Slide Time: 22:42)



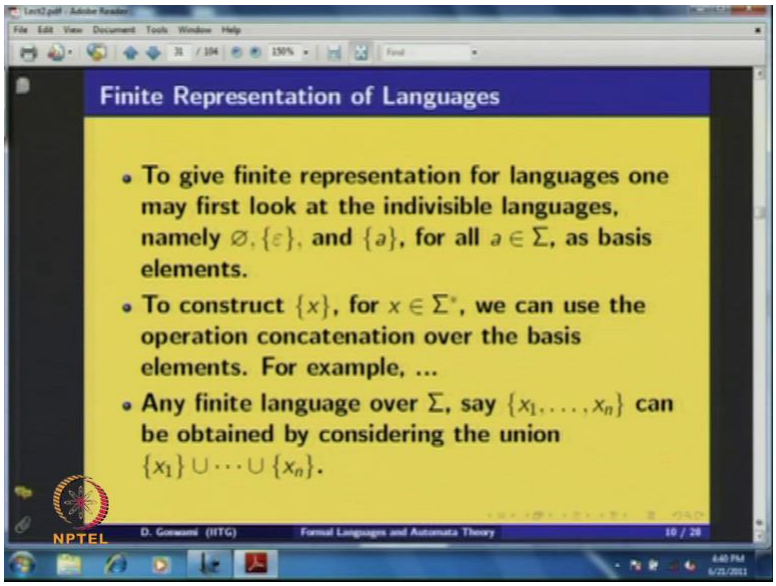
Therefore, the giving finite representation for infinite languages is a nontrivial problem. The language is finite you can over any more dustings and which could be a finite representation for the languages. So, let see how to give a finite representation for in infinite languages. So, in this context the operations on languages may be helpful. Let see that various operations that we have discussed so far, the views to be discarded from representation. For example, using Kleene star operation we can have finite representation for some infinite languages.

(Refer Slide Time: 23:34)



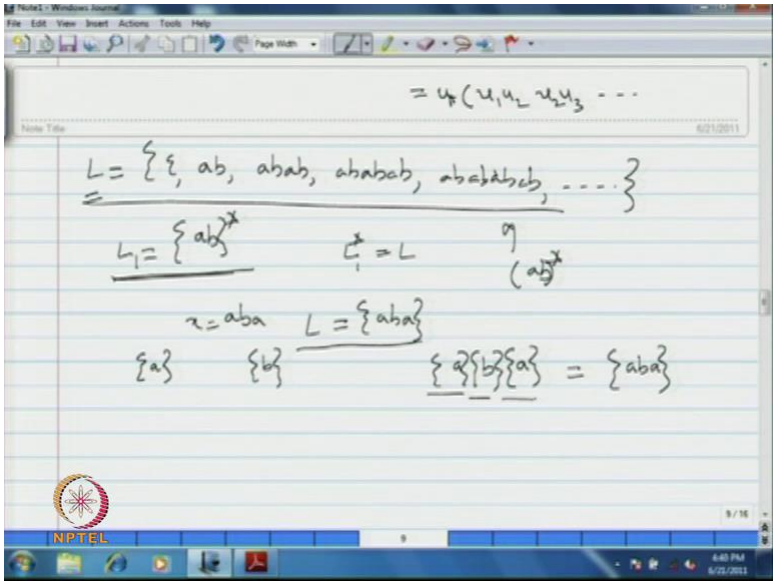
For example, we consider the language say the language L which is the empty string epsilon then suppose a b, a b a b, a b a b a b, a b a b a b a b and so on, that means the repetition of strings concatenate a or concatenating a b with a b finitely many times. Now, you can find this languages or this languages can be constructed as I Kleene to z or the language say a b, that means you consider a language a b and take a Kleene to the width we get this languages are suppose to L 1. We see that L 1 star exactly L, that means this infinite language can be represent as a Kleene closure of this languages L 1. Now, if you have this if we represent the language as simply a b, then Kleene closure of this is nothing but the language L. So, that is how it represent an infinite languages using finite representation.

(Refer Slide Time: 24:56)



Now, to give finite representation for languages one may first look at the individual languages namely, the phi singleton epsilon and a, because we cannot divide those languages further, these are the basis elements. The singleton a for every a belong to sigma, they are all basic elements. Suppose, we want to construct the languages containing the singleton x for some string x belong to sigma star. We can use the operation concatenation over the basis elements.

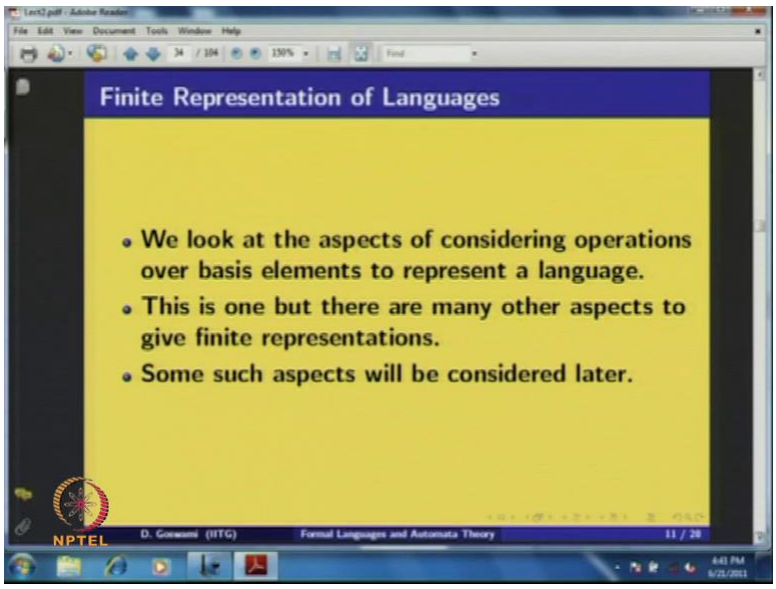
(Refer Slide Time: 25:38)



Just is the example, suppose you have x equal to a b that means the languages a b containing the string only a b L, so this is the language L. So, what you can do is that, you can consider a language containing singleton a is a basis element, then singleton b then we concatenate in this language a and b we get a language containing only just a misting a b, then again concatenate with a. So, this you give the languages containing the single string a b a.

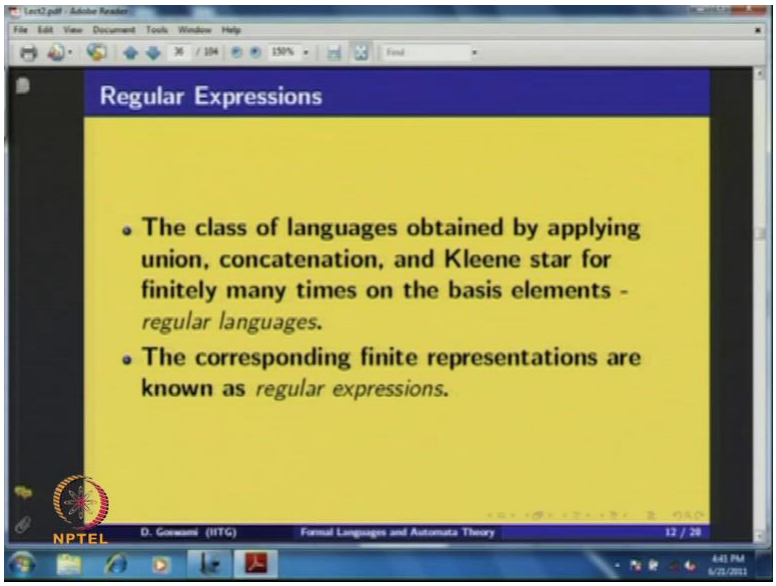
Therefore, this language can be constructed by taking concatenation of 3 languages which are the basis elements. Any finite language over sigma say x 1, x 2 up to x n, it is x i jesting over sigma star can be obtained by considering the union of this singleton elements x 1 union x 2 and so on, that means you can consider the operations Kleene closure concatenation, and ((Refer Time: 27:04)) to apply on the basis elements to construct any kind of languages.

(Refer Slide Time: 27:10)



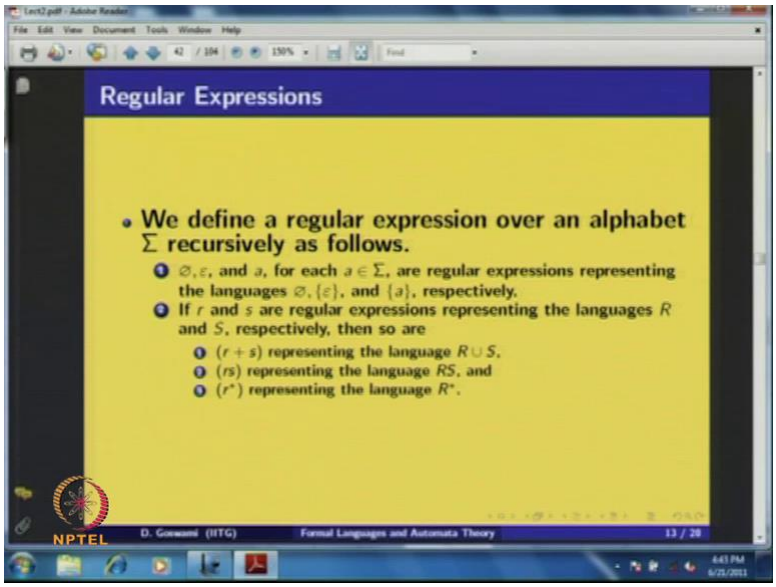
Now, we look at the aspects of considering operations over basis elements to represent a language. This is one of the aspects, but there are many other aspects to give finite representation. We will consider over the aspects later on.

(Refer Slide Time: 27:36)



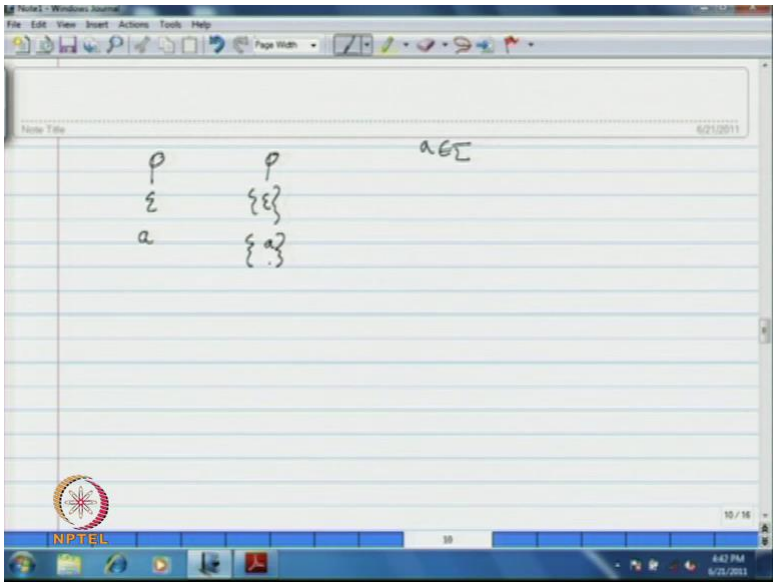
Now, the class of languages that we get by applying union, concatenation, and Kleene to Kleene closure for finitely many times on the basis elements is known as regular languages. The corresponding finite representations are known as regular expressions.

(Refer Slide Time: 27:58)



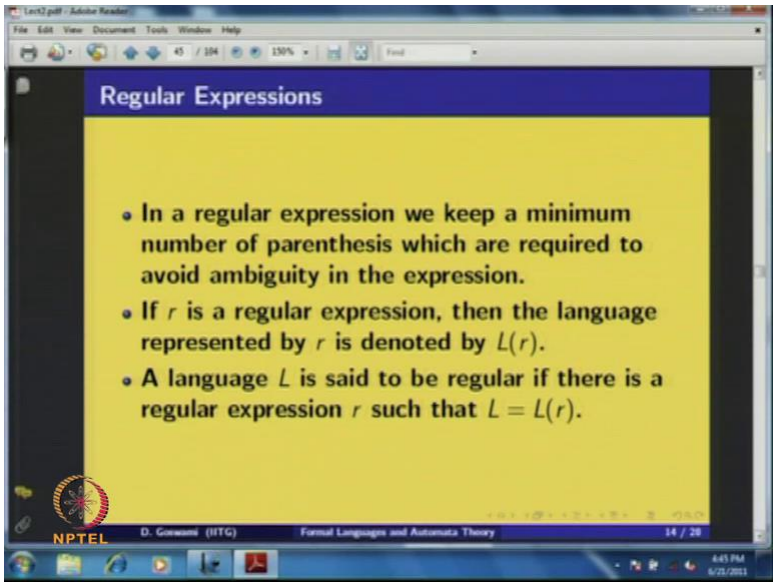
Now, let us define regular expression over an alphabet sigma recursively as follows. We consider phi, epsilon, and a for every a belongs to sigma to the regular expression, representing the languages phi, the singleton epsilon, and a respectively.

(Refer Slide Time: 28:27)



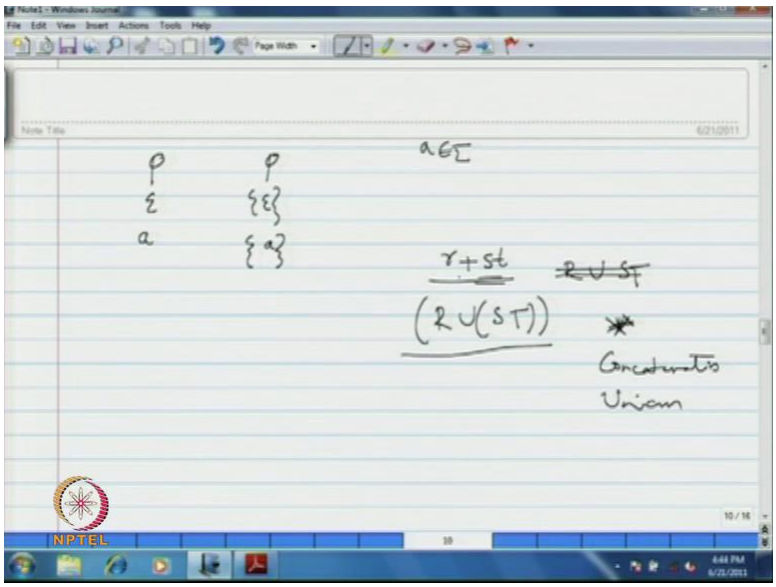
That means, if we have phi if we said as a regular expression, and this represented a language phi. If we have the regular expression sigma epsilon, then which represents the language contains the singleton epsilon. And for every element a, a belong to sigma for any element I will in the sigma a is a regular expression, and it denotes or represents a language containing a single string of Lang to 1 which is a itself. So, this the basis case for a definition of regular expressions. Now, if r and s are regular expressions representing the languages capital R and capital S respectively. Then so are the following, r plus s representing the language R union S, then r concatenation s or simply r s representing the language R concatenation S, and r star representing the language capital R star.

(Refer Slide Time: 29:44)



In a regular expression we keep a minimum number of parentheses which are required to avoid ambiguity in expressions.

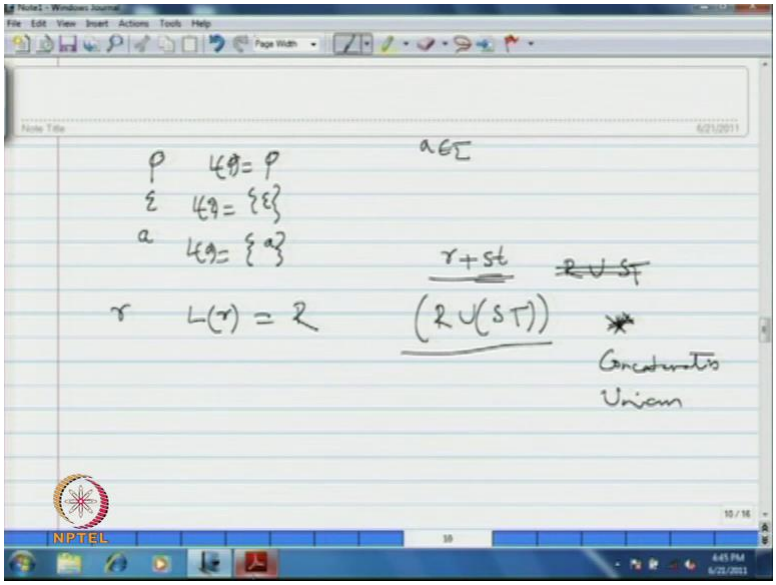
(Refer Slide Time: 30:00)



For example, if r plus s t the regular expression then actually this represent the languages R union S concatenation T. So, in this case we have some precedence to normally Kleene closure has more precedence hash precedence, then we have the precedence for concatenation and then for union. Therefore in this case, so this language can be written by the regular expression r plus s t, because in this case the concatenation has hash

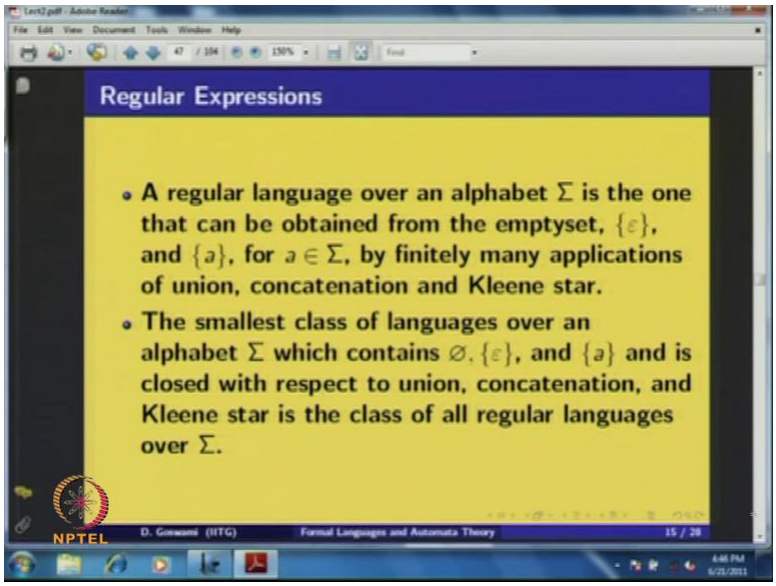
precedence, and then we had precedence for union. And if r is a regular expression, then the language represented by r is denoted by L (r).

(Refer Slide Time: 31:15)



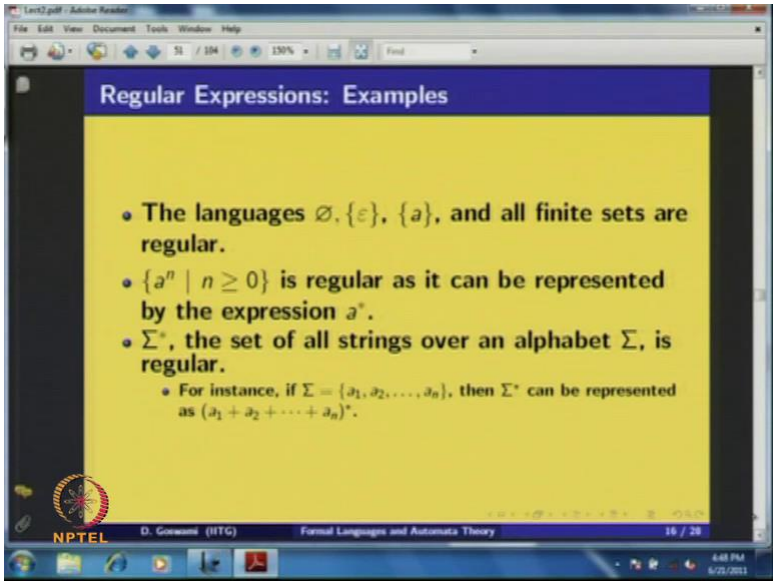
So, if phi the regular expressions then we write that L phi, the language written by phi is phi. Similarly, epsilon is the regular expression L of epsilon is the singleton epsilon similarly L of a is a itself. Similarly, if r is the regular expression the corresponding regular expression L (r) may be some set capital R and so on. And a language L is said to be regular, if there is a regular expression r such that L equal to L (r), because we be the regular expression by applying finitely many operations from union concatenation and Kleene closure of order basis element, and that is have it defined a regular languages.

(Refer Slide Time: 32:08)



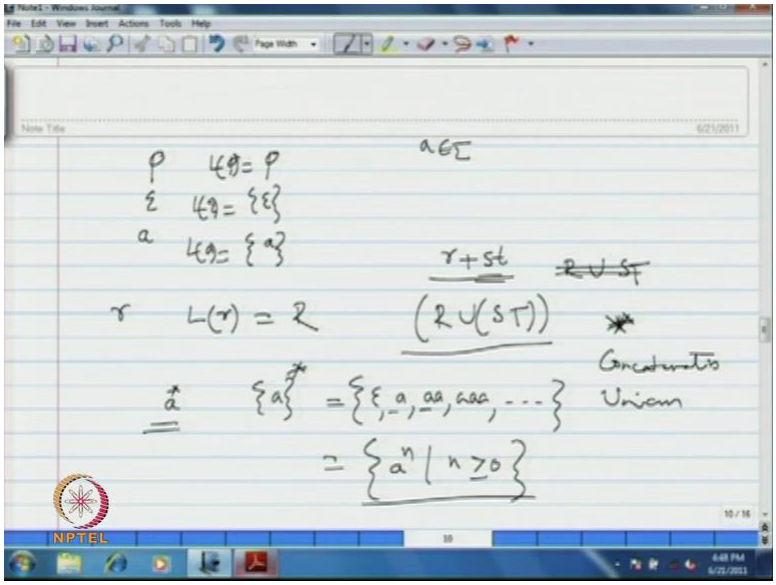
So, a regular language over an alphabet sigma is the one that can be obtained from the empty set singleton epsilon, and a for singleton a for every element to sigma by finitely many applications of union, concatenation, and Kleene closure. And the smallest class of languages over an alphabet sigma which contains phi, singleton epsilon, and singleton a, and is closed with respect to union, concatenation, and Kleene closure is the class of all regular languages over sigma. So, this can be same from the definition.

(Refer Slide Time: 32:46)



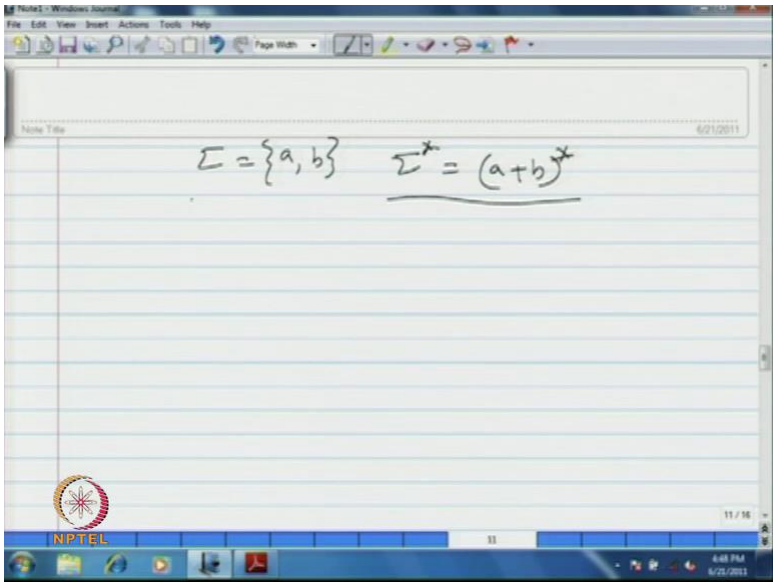
Now, let us give some examples of regular expressions, and how can be construct regular expressions. So, already we have see that the language phi, singleton epsilon, and singleton a for every element sigma, they are all finite sets and are regular. Consider a to the n for n greater than equal to 0. So, this set, this set is regular as it can be written by the regular expression a star because.

(Refer Slide Time: 33:19)



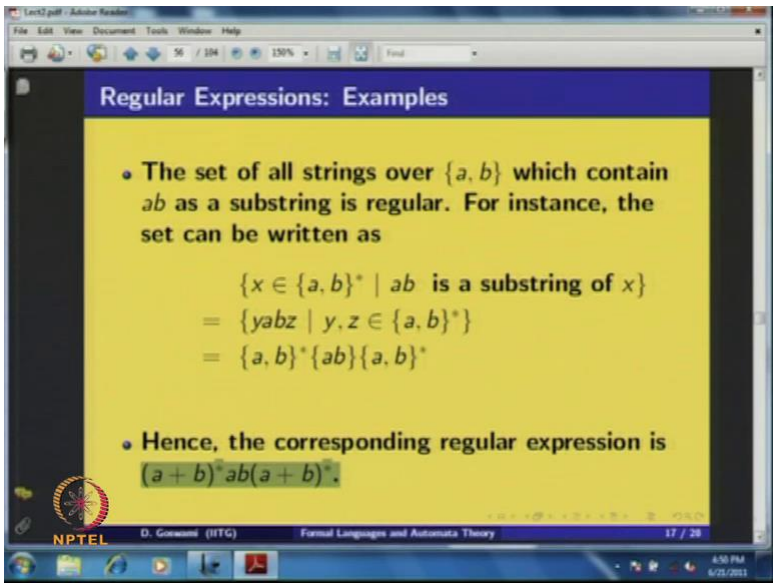
So, a star basically a represents applying a Kleene closure to the language a. So, if we applied is Kleene closure to this a, the language containing a singleton a, then we get epsilon a, a a, a a a and so on. So, this nothing but a to the n, n greater than or equal to o 1 equal to 0 will get a string, and equal to 1 get a, and then be 2 you get 2 a a and so on. So, therefore a to the n can be denoted by using a star, and hence this set is a regular set. Similarly, sigma star a set of all strings over an alphabet sigma is regular. For instance, if sigma is the set containing a 1, a 2, up to a n, then sigma star can be written as a 1 plus a 2 plus a 3 plus a n whole star.

(Refer Slide Time: 34:42)



For example, if sigma equal to say a b then sigma star equal to a or b whole star, so we can define like this, therefore sigma star is a regular expression.

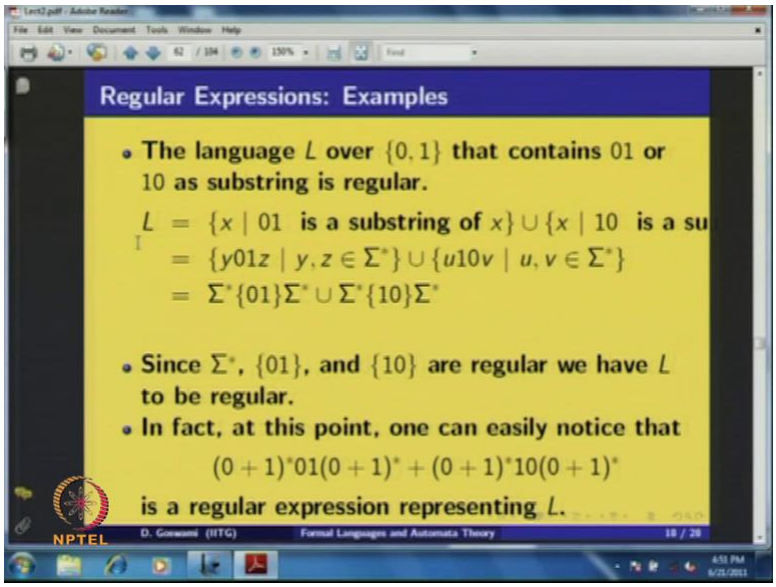
(Refer Slide Time: 35:05)



Thus consider the language, the set of all strings over a, b which contain a b as a substring, I can show that this set is a regular set. For instance, you can write this set as all those strings x belong to a b star subset a b is a substring of x, and then this can be written as y a b z for some y z belong to sigma star. So, here y and z may be any string from a b. So, any string from a b can be written as a b whole star, and this is a b whole

star. So, we can write this is a concatenation of 3 languages this a b, a b star, and a b star. Now, this can be written as a plus b whole star a b a plus b star, so these are regular expression for that v 1 language, and hence this set is regular.

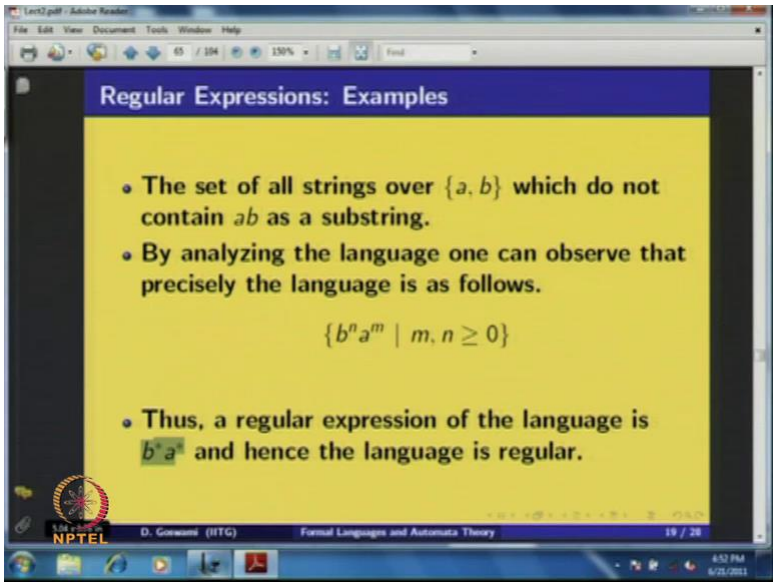
(Refer Slide Time: 36:15)



Thus consider language L over 0 1 that contains 0 1 or 1 0 as substring. This can also be some to be regular, because it write it as the set of all strings x such that 0 1 is substring of x union all strings over x such that 1 0 is substring of x. Now, it can be written as some y 0 1 z for some y z belong to a sigma star union u 1 0 v for some u v belong to sigma star is nothing but sigma star 0 1 sigma star, because y z may be anything from sigma star union sigma star 1 0 sigma star, because u v be anything from sigma star.

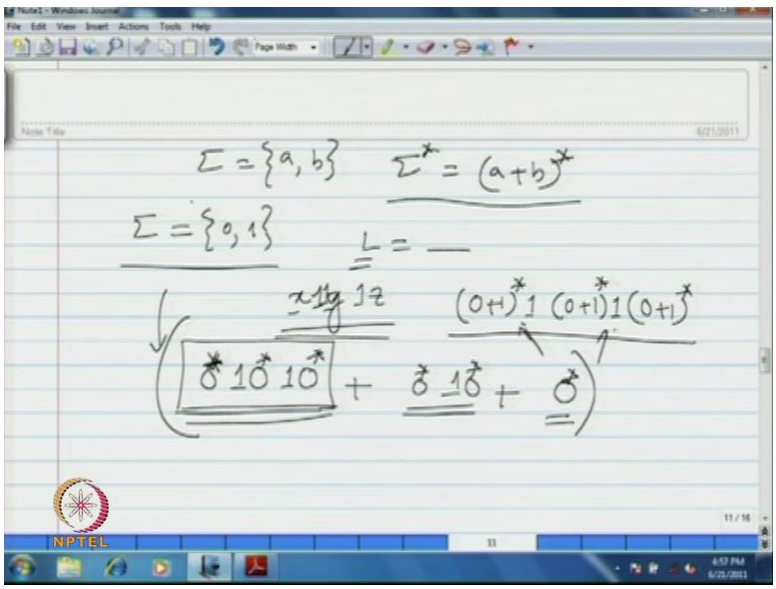
Therefore, this is regular expression, because sigma star 0 1 and 1 0 are regular, and we have express this language L using operations concatenation, Kleene closure, concatenation, and union over the regular sets sigma star 0 1 and 1 0. Therefore, this L must be a regular set. So in fact, we can write it as 0 plus 1 star minus sigma star 0 1 0 plus 1 star plus which ((Refer Time: 37:53)) union 0 plus 1 star 1 0 (0 plus 1) star. So, it is a regular expression representing that v 1 language L.

(Refer Slide Time: 38:04)



Similarly, the set of all strings over a b which do not contain a b as a substring, one can really see that this commutation has b to the n a to the m for some m n greater than equal to 0, because a b cannot occur a substring. Therefore, if any b occurs it must precede any occurrence of b (s). Therefore, this can be written as a regular expression which is nothing but which is b star a star. Therefore, this language the set of all strings over a b which not containing a b substring can be written by the regular expression b star a star, and hence this is a regular language. Similarly, we can construct regular expressions for many other languages.

(Refer Slide Time: 38:59)



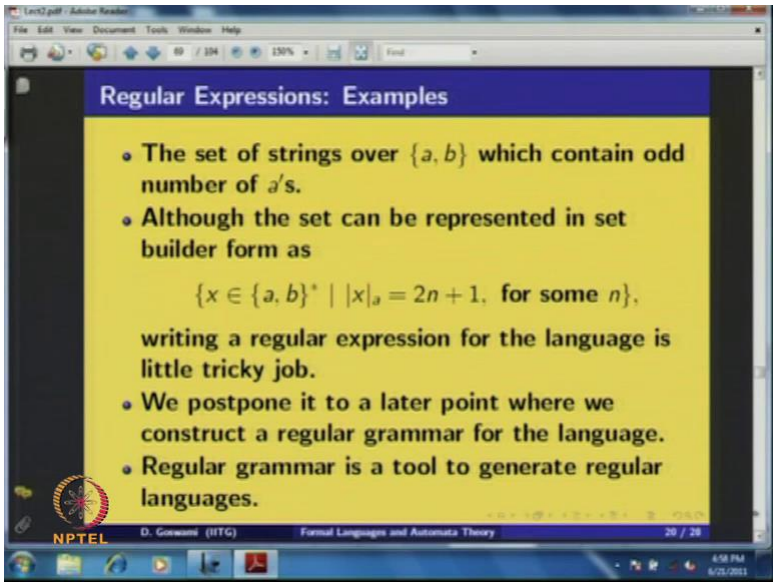
For example, just consider the alphabet 0 1 a set of all strings over this alphabet where number of say 1s is at least 2, that means there are at least 2 occurrence of 1 in every string the language, thus define a languages like this, the set of all strings containing over 0 1 where the numbers of wants as at least 1. So, since at least 2, so since 1 must occur at least 2 times. Before that did not we any string over 0 1 written by x, we could a first consider 1 1, after that also then any string over 0 1 is defined as y. Then the second 1, and after the second 1 again we have string about 0 1 which contain n number of 0s and 1s.

So, therefore the typical string the language can be written by x 1 y 1 z, and here x y z may be any string over 0 1. Therefore, the corresponding regular expression be 0 plus 1 star and this could be x, then 1 again 0 plus 1 star again 1, this 0 plus 1 star represents is y, and finally for z again 0 plus 1star. So, this 1 and this 1 represents that at least 2 1s will be there in any string, and other than these 2 language any string so far 0 1 in any represents, so this regular expression for the given languages L. Similarly, if we defined a languages over the same alphabets, suppose set of all strings so for 0 1 having at most 2 occurrences of 1.

So, this one 1s for at least 2 occurrence of 1s at least 2 occurrence, but now the for the languages while we have say at most 2 occurrence of 1s. In such a case there can be at most 2 occurrences of 1s. So, therefore, before is 1 then will be any string of over 0s and 1s or any string of 0s, but there cannot be any 1. After this second may have any strings of 0s, and then we have it is again 1 more 1 and occurs that 1 so we have any occurrence of 0s strings of 0s. So, there will be 2 occurrences of 1s, and this is the string for exactly 2 occurrences of 1s, the set of all strings of 0 1 which have exactly 2 occurrence of 1s.

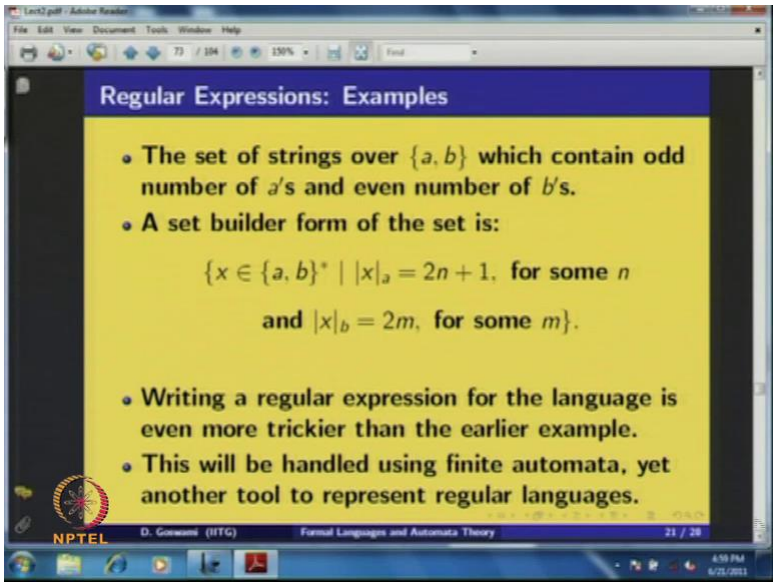
So, if you should have at most 2 occurrence of 1s, then should ((Refer Time: 42:35)) case where the strings of the form where at least at most 1 1 0 star 0 star, and only strings of 0s. So, this ((Refer Time: 43:00)) case, but there is no 1s which regular expression covered a case where there is only 1 1, and this in ((Refer Time: 43:09)) case where there is at most 1s. So, this may the union of all this 3. Hence, since we can express it using this regular expression, therefore the languages is a regular one.

(Refer Slide Time: 43:28)



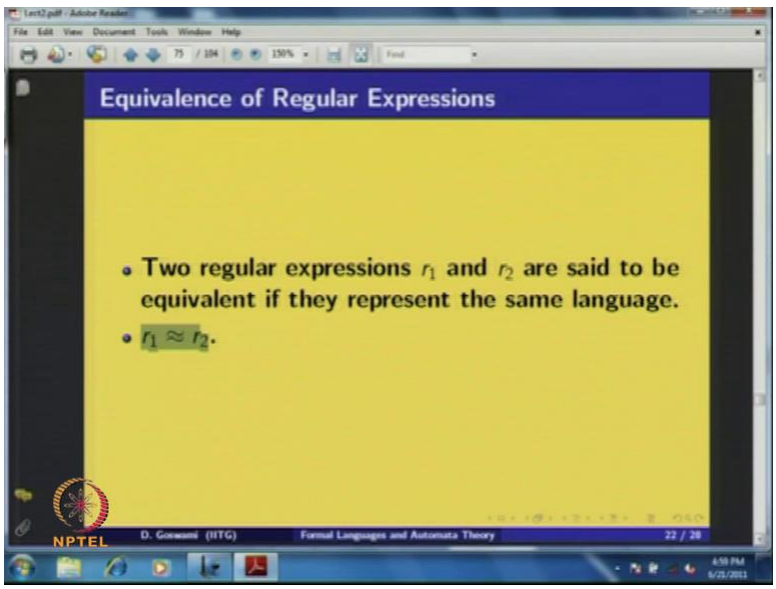
Now, just consider a language a set of strings over a, b which contain odd number of a (s). Now, it is b to see that, it can be represented as set builder form x belong to a, b star such that number of ((Refer Time: 43:44)) of a (s) in x physical to twice n plus 1 for some n. But, writing a regular expression for this language is little bit tricky. So, we postpone it to a later point where we construct a regular grammar for the language. So, regular grammar is under representation finite expression for a language. So, even though it is tricky to at regular expression is very easy to write a regular grammar or construct a regular grammar for the given language. So, regular grammar is a tool to generate exactly the write class over regular languages.

(Refer Slide Time: 44:17)



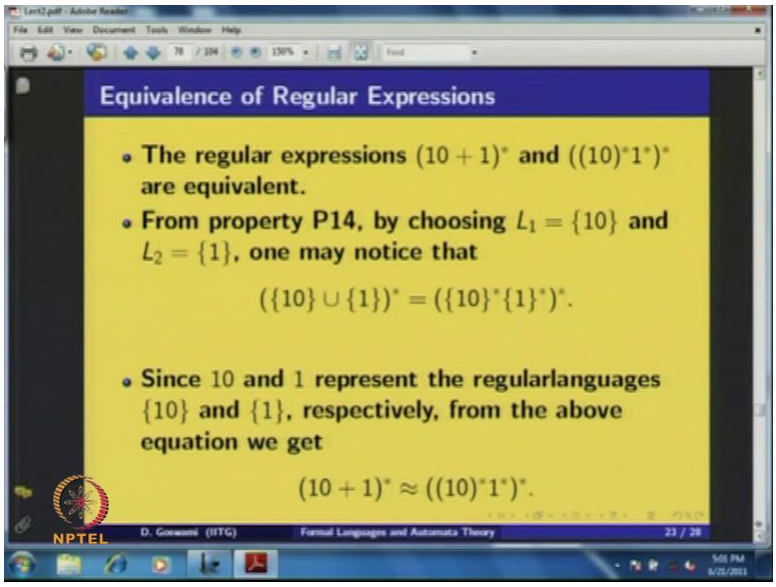
Then again consider the set of all strings over a, b which contain odd number of a (s) and even number of b (s). Again we can write this set, a set builder form is x belong to a, b star such that numbers of a (s) in x equal to twice n plus 1 ((Refer Slide Time: 44:35)) for some n, and numbers of b is in x equal to twice m for some m. Writing a regular expression for this languages more trickier than the previous example. And we can use some other tool like say finite automata to construct this kind of auto except this kind of languages is again finite automata is yet again another tool to represent regular languages.

(Refer Slide Time: 45:05)



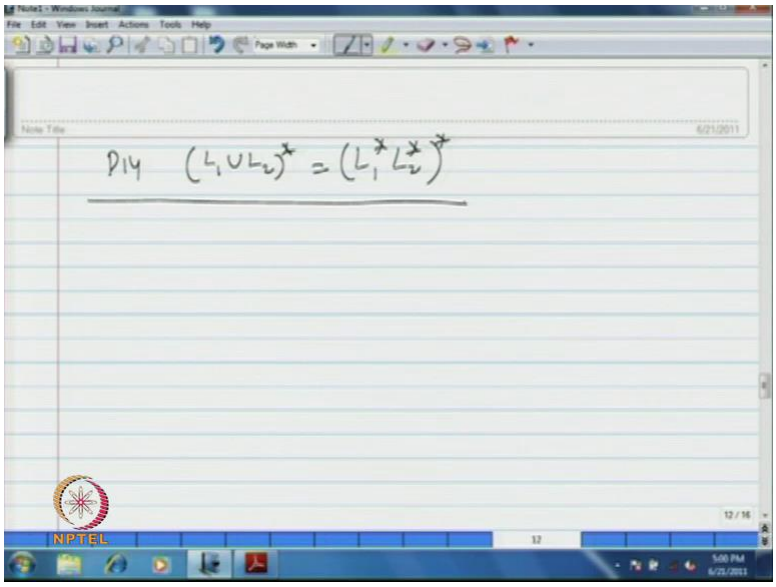
Now, let see the equivalence of regular expressions, you say that 2 regular expressions r 1 and r 2 as equivalent if they represent the same language. And we denoted like r 1 equivalent to r 2, and use this symbol to represent the equivalence of 2 regular expressions. So, this means that L of r 1 is equal to L of r 2.

(Refer Slide Time: 45:35)



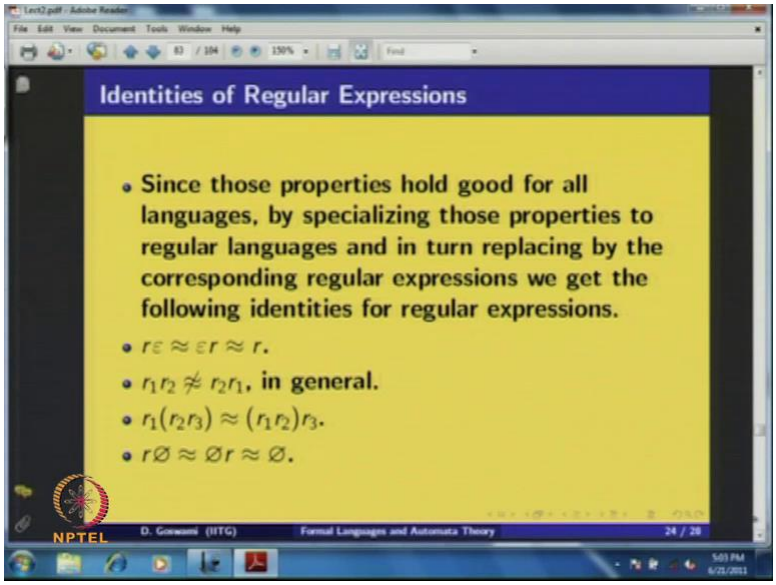
Let us consider the regular expression 1 0 plus 1 whole star, and 1 0 star 1 star star, which can so that this 2 regular expressions are equivalent.

(Refer Slide Time: 46:09)



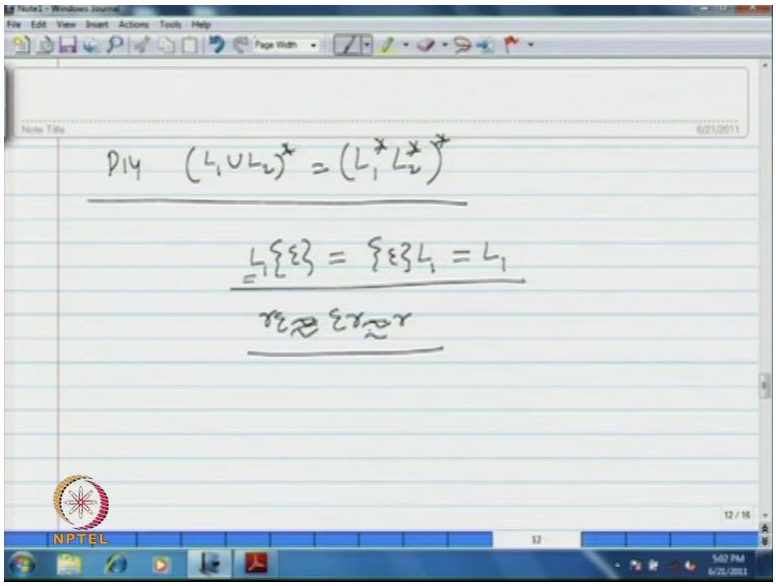
From property 14 that we have already discussed regarding a property of languages is nothing but L 1 union L 2 star equal to L 1 star L 2 star whole star. So, it is a property 14 that we have already discussed for languages. Say if you consider property 14 then assuming L 1 to be 1 0, and L 2 to be 1 we get exactly 1 0 union 1 whole star to be equal to 1 0 star 1 star whole star. Now, since 1 0 and 1 represent the regular languages 1 0 singleton 1 0 and singleton 1 respectively. From the above equation we get that 1 0 plus 1 whole star is exactly is equivalent to 1 0 star 1 star whole star. So, you can use the property of languages to so that the corresponding regular expressions are equivalent.

(Refer Slide Time: 47:15)



That means these properties holds good for all languages. Since the since those properties hold good for all languages by specializing those properties to regular languages, and in turn replacing by the corresponding regular expressions we get the following identities for regular expressions, r epsilon is equivalent to epsilon r is equivalent to r.

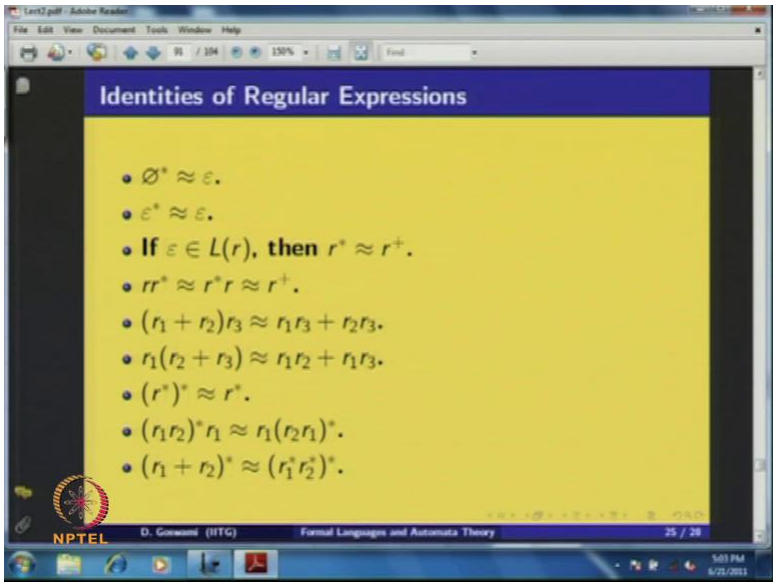
(Refer Slide Time: 47:45)



That means we have considered the property L 1 concatenation into epsilon is equivalent to or equal to epsilon concatenation with L 1 which is exactly L 1. So, if L 1 suppose r ((Refer Time: 48:05)) regulation for L 1, and epsilon are regulation for the singleton epsilon, this is equivalent to singleton epsilon and the corresponding regulation for along it is r is equivalent to r. So, these equivalents have got from the properties of the language.

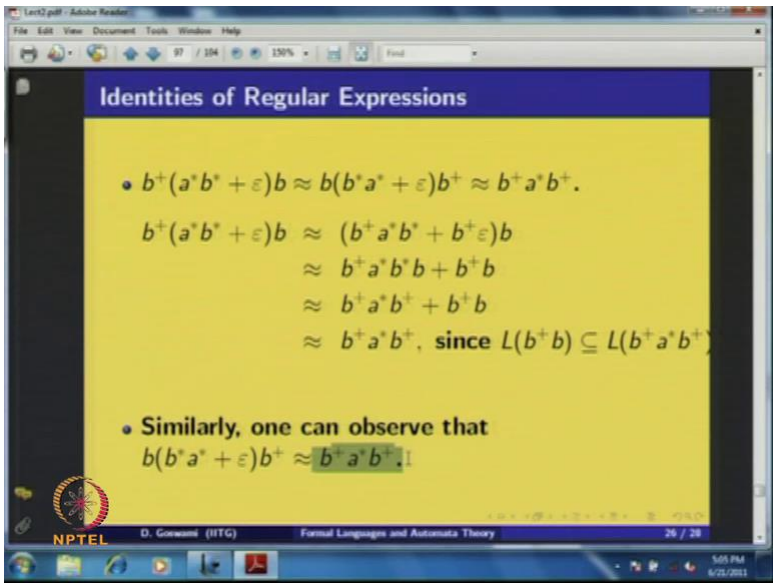
Similarly, by replacing the language by regular expression in the properties of languages, we get different kinds of equivalents for regular expressions. For example, say r 1 r 2 is not equivalent to r 1 r 2 r 1 in general. Similarly, r 1 concatenation r 2 r 3 not equivalent to or equivalent to r 2 r 1 r 2 concatenation r 3, r phi is equivalent to phi r is equivalent to phi.

(Refer Slide Time: 49:02)



Phi star is epsilon and so on. We can just reproduce these properties from the basic questions of regular expressions from the properties of languages.

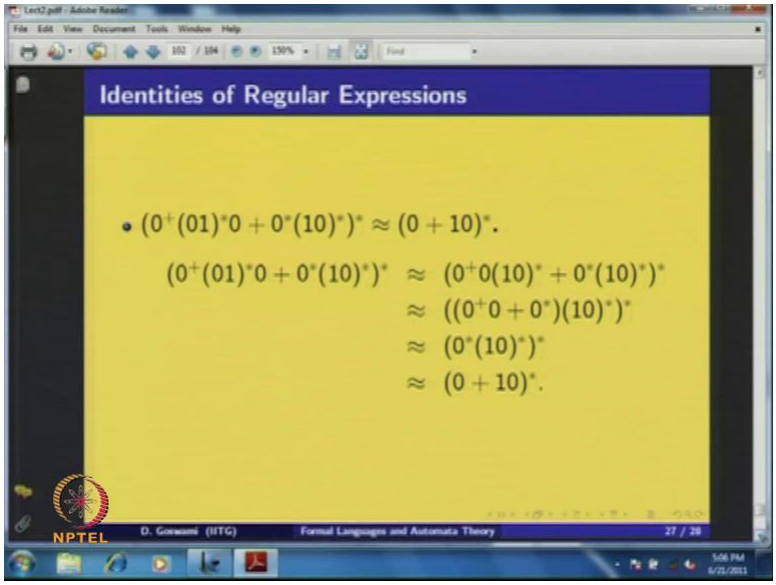
(Refer Slide Time: 49:22)



Just consider this identity of regular expressions, you can use or prove this identity by using the equivalence of regular expressions. For example, say b plus a star b star plus epsilon b is equivalent to b plus a star b star plus b plus epsilon b is equivalent to b plus a star b star b plus b star b. So, in this case simply this b plus we have concatenate established and concatenate epsilon I have got this result. Similarly, this b is concatenate

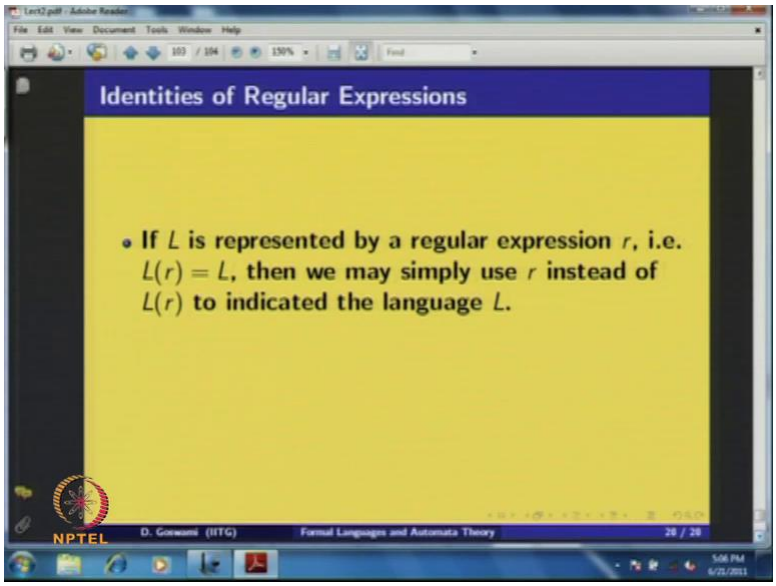
this term and this term. Now, this can be written as b plus a star b plus, hence this can be written as b plus a star b plus, because b plus b a subset of b plus a star b plus. Therefore, we have got the first one, b plus a star b star plus epsilon b is equivalent b plus a star b plus this will go. Similarly, one can observe that b b star a star plus epsilon b plus is equivalent to b plus a star b plus.

(Refer Slide Time: 51:13)



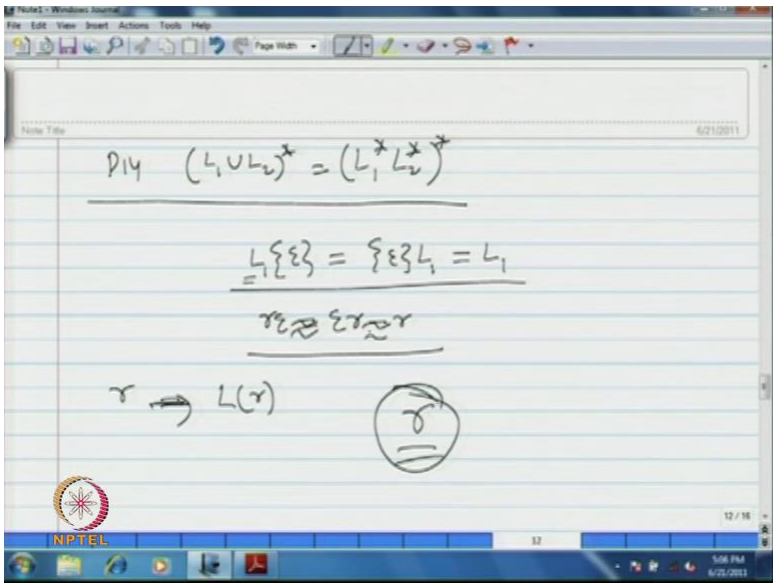
Similarly, we can so that these 2 regular expressions are equivalent by using reference step. So, the first step we have got from this expression, this equivalence expression from this we can use the regular expression properties to get eventually this expression, and finally this can be written as 0 plus 1 0 star.

(Refer Slide Time: 51:48)



Now, if L is represented by a regular expression r, that is L (r) equal to r, then we may simply use r instead of L (r) to indicate the language.

(Refer Slide Time: 52:05)



So, for given L language say L (r), it is r the regular expression we know that L (r) is the corresponding language written by r, but simply we can some has write r to represent a language itself. So, it was ((Refer Time: 52:22)) notation for different a language to for language written by ((Refer Time: 52:27)) equation r.