Discrete Mathematical Structures

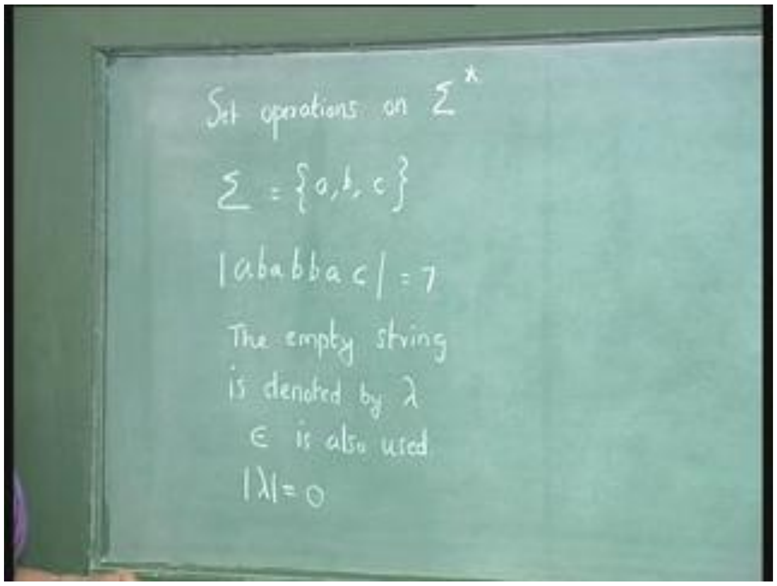
Dr. Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture -12

Set Operations on Strings over an Alphabet

Today we shall consider set operations on sigma star where sigma is an alphabet. So let us take an alphabet sigma for example say a, b, c. The set of strings over sigma is denoted as sigma star and we want to consider operations on the set of strings over sigma star. What is a string over sigma? For example say ababbac this is a string over sigma. It is a sequence of symbols taken from sigma and you denote the length of this string this ababbac is a string over sigma. The length of this will be 7. So you denote the length of the string like this. The empty string is denoted by lambda. Usually it is denoted by lambda but some times epsilon is also used.

(Refer Slide Time: 2.34min)



Set operations on Σ^*

$\Sigma = \{a, b, c\}$

$|ababbac| = 7$

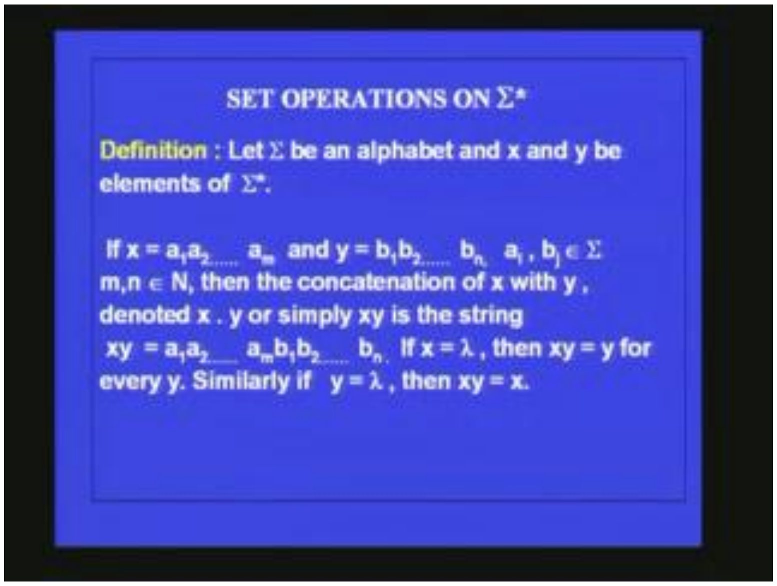
The empty string is denoted by λ

ϵ is also used

$|\lambda| = 0$

The length of the empty string is 0. If you take the length of a it is 1, if we take the length of ab it will be 2 and so on. Now we want to define some operations on this. The operation to be defined on the strings is first concatenation. Let us see what concatenation is. Let sigma be an alphabet and x and y be elements of sigma star. As I told you sigma star is the set of all strings over sigma including the empty string. If you have one string x is equal to a_1 a_2 a_m and another string y is equal to b_1 b_2 b_n where all these symbols are from sigma, then the concatenation of x with y is denoted by x dot y or simply xy is the string xy is equal to a_1 a_2 a_m and b_1 b_2 b_m. If x is equal to lambda or the empty string then xy will be equal to y for every y. Similarly, if y is equal to lambda then xy will be equal to x.

(Refer Slide Time: 3.03)



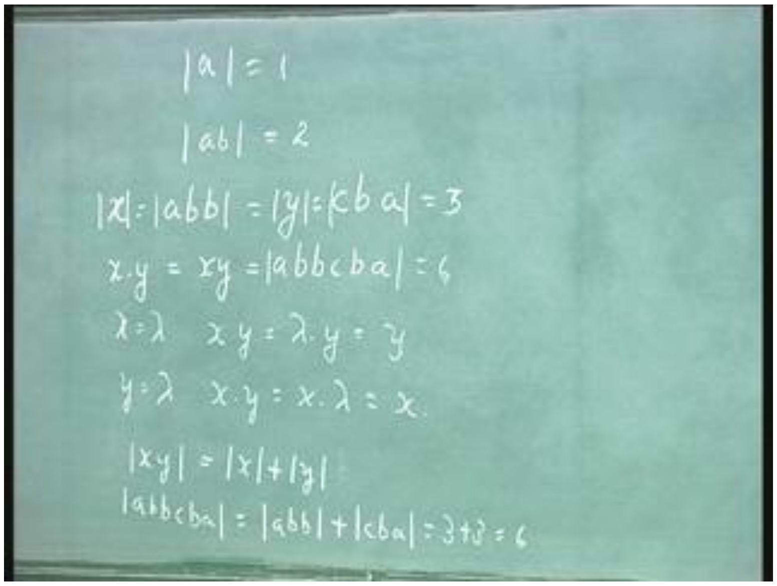
SET OPERATIONS ON Σ^*

Definition : Let Σ be an alphabet and x and y be elements of Σ^*.

If $x = a_1a_2 \ldots a_m$ and $y = b_1b_2 \ldots b_n$, $a_i, b_j \in \Sigma$, $m,n \in N$, then the concatenation of x with y, denoted x . y or simply xy is the string $xy = a_1a_2 \ldots a_mb_1b_2 \ldots b_n$. If $x = \lambda$, then $xy = y$ for every y. Similarly if $y = \lambda$, then $xy = x$.

Let us take some example and see suppose I have a string x, x is equal to say abb and y is equal to cba. Then what is the concatenation of x with y? You can write it as x dot y. Usually is it is written like xy only. It is, the symbols in x that is abb and then symbols in y are placed by the side of x that is cba. The concatenation of x with y is denoted by abbcba. Now if x is equal to lambda then xy will be just lambda into y that will be y and if y is equal to lambda xy will be x into lambda that is x. So if you concatenate with the empty string on the right or on the left you get the same string. So what can you say about the length of xy? The length of xy in this case we see the length is 6, the length of this is 3. So the length of xy will be the length of x plus length of y. In this particular example it is 3 plus 3. So the length of abbcba is the length of abb plus length of cba that is equal to 3 plus 3 you can see it is 6. So that also confirms to this, length of x plus length of y will be length of x plus length of lambda is 0 that is equal to the length of x.

(Refer Slide Time: 6.00min)



$|a| = 1$

$|ab| = 2$

$|x| = |abb| = |y| = |cba| = 3$

$x.y = xy = |abbcba| = 6$

$x = \lambda \quad xy = \lambda.y = y$

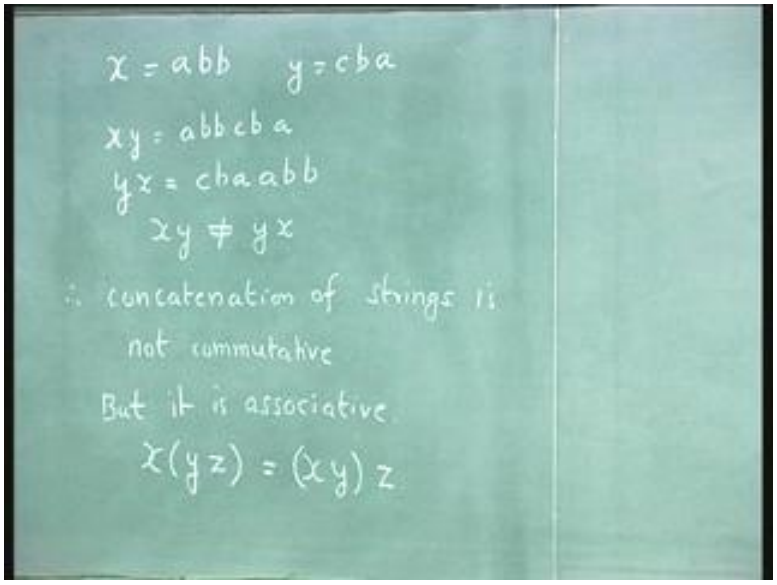
$y = \lambda \quad x.y = x.\lambda = x$

$|xy| = |x| + |y|$

$|abbcba| = |abb| + |cba| = 3+3 = 6$

Now in this example which you have taken I will repeat that x is equal to abb, y is equal to cba. So xy is equal to abbcba and yx is equal to cbaabb. We can very easily see that xy is not equal to yx. Therefore concatenation of strings is not commutative but it is associative. That is you can see that x concatenated by yz is equal to xy concatenated with z.

(Refer Slide Time: 7.36)



$x = abb \quad y = cba$

$xy = abbcba$

$yx = cbaabb$

$xy \neq yx$

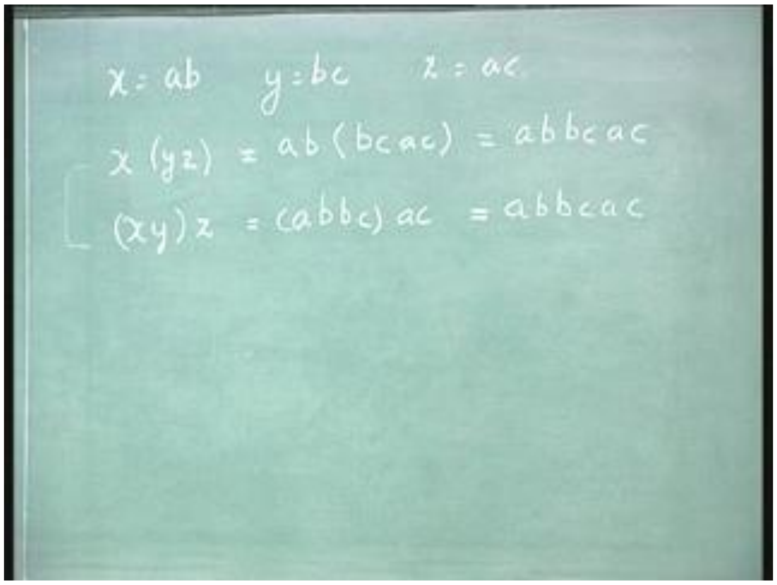
$\therefore$ concatenation of strings is not commutative

But it is associative

$x(yz) = (xy)z$

This you can very easily see. Again let me take an example, x is equal to ab y is equal to bc z is equal to say ac, then what is x concatenated with yz that is ab concatenated with yz is bcac that is equal to abbcac. Now what is x concatenated with y and then concatenated with z? xy will be abbc concatenated with z is ac that will again give you abbcac. So concatenation is an associative operator, you can these you can see that these two are equal. It is not commutative but it is associative.

(Refer Slide Time: 8.38)



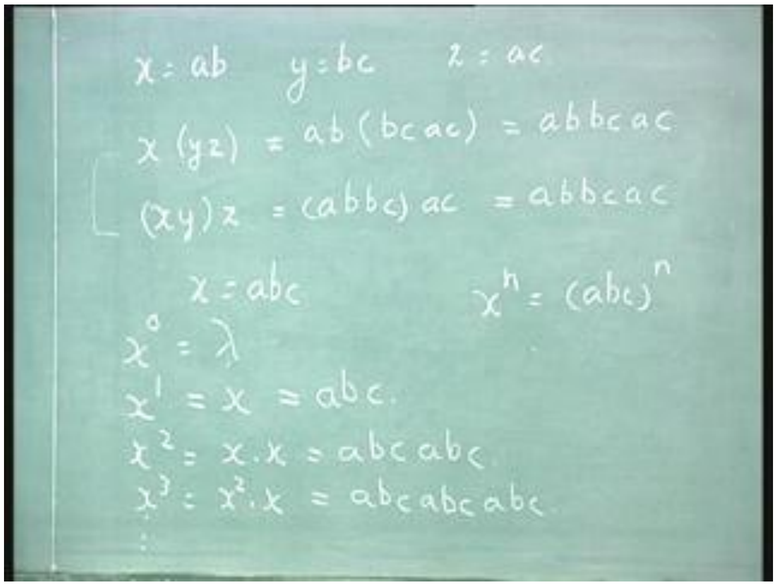
$x: ab \quad y: bc \quad z: ac$

$x(yz) = ab(bcac) = abbcac$

$(xy)z = (abbc)ac = abbcac$

Now you can also define some thing like x power 0. Let us see what it is. Let x be an element of sigma star, for each n belonging to N the string x power n is defined as follows: x power 0 is lambda and x power n plus 1 is x power n into x, because of the associative property of concatenation we can define this very easily. Take any string x say x is equal to abc then x power 0 is denoted as lambda which is the empty string, x power 1 is x itself that is abc, x square is x concatenated with x that is abc abc, x cubed will be x squared concatenated with x. You can write this without any ambiguity because of the associative property. And that will be abcabcabc. In general you can note that x power n will be abc power n and abc abc abc written n times. So for a string we can define the power in this manner.

(Refer Slide Time: 10.19)



$x : ab \quad y : bc \quad z : ac$

$x(yz) = ab(bcac) = abbcac$

$(xy)z = (abbc)ac = abbcac$

$x = abc \qquad x^n = (abc)^n$

$x^0 = \lambda$

$x^1 = x = abc.$

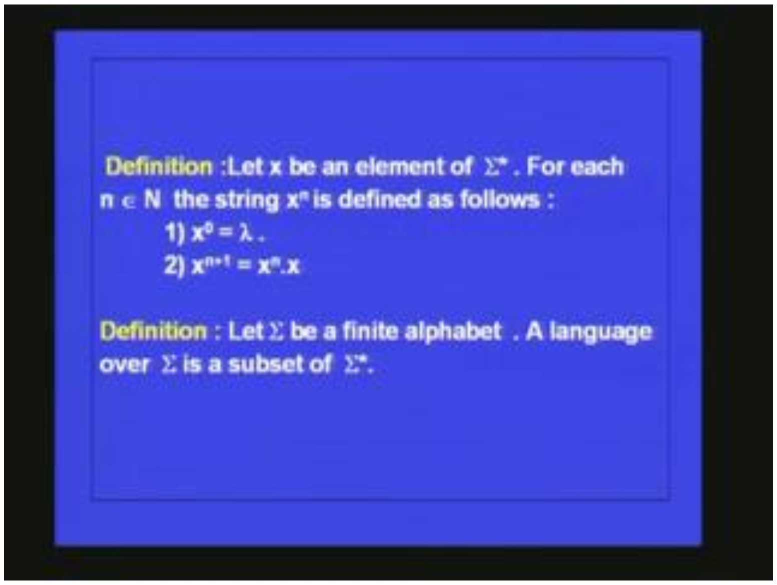
$x^2 = x.x = abcabc$

$x^3 = x^2.x = abcabcabc$

$\vdots$

Let sigma be a finite alphabet, then what do you say about the language over sigma? A language over sigma is a subset of sigma star. So, after defining concatenation and a power of a string we see what an alphabet is. Usually we talk about languages over an alphabet. So a language is defined in this manner. A language over sigma is a subset of sigma star.

(Refer Slide Time: 10.20)



Definition :Let x be an element of Σ^* . For each $n \in N$ the string x^n is defined as follows :

1) $x^0 = \lambda$.
2) $x^{n+1} = x^n.x$

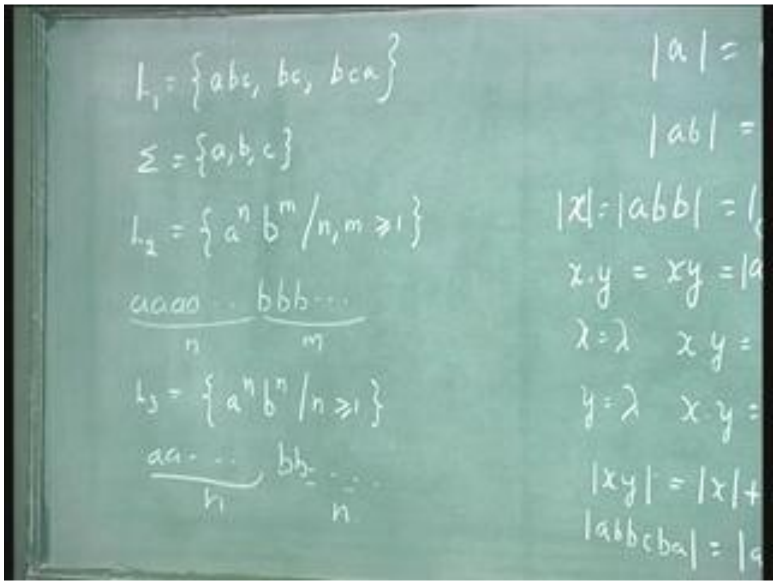
Definition : Let Σ be a finite alphabet . A language over Σ is a subset of Σ^*.

Consider for example, L_1 equal to abc bc bca where the alphabet is taken as a comma b comma c. This is a subset of sigma star that is you are taking some strings from sigma star and L_1 has these strings, it is a finite set. So L_1 is a finite language over sigma as it consists of a finite subset of sigma star. This has got three strings. You can define infinite subsets as well.

For example, you can consider some thing like this L_2 equal to a power n b power m n comma m greater than or equal to 1. So what sort of strings will belong to L_2? a power n that is n a's followed by m b's where n and m are integers greater than or equal to 1. So L_2 is an infinite subset of sigma star and it consists of strings of the form as a sequence of a's followed by a sequence of b's. No other string is in this language, it has only strings of this form, string of a's or a sequence of a's followed by a sequence of b's. So this is another example. Take another example L_3 equal to a power n b power n and n greater than or equal to 1.

Again this is an infinite subset of sigma star. This language consists of strings of the form aaa n a's followed by an equal number of b's. This is also of the same form but here n and m can be different, here the sequence of a's should be followed by an equal number of b's, that is string of a's followed by an equal number of b's n and n, and n can vary from 1 2 3 etc, it can be any integer. So this is again another infinite subset of sigma star. This is another language. Later on if you study automata and formal language theory you will learn that this is a finite set, this is a regular set and this is a contextary language, this is not a regular set and this is a poset and infinite set.

(Refer Slide Time: 14.05)



$L_1 = \{abc, bc, bca\}$

$\Sigma = \{a, b, c\}$

$L_2 = \{a^n b^m / n, m \geq 1\}$

$\underbrace{aaaa\cdots}_{n}\ \underbrace{bbb\cdots}_{m}$

$L_3 = \{a^n b^n / n \geq 1\}$

$\underbrace{aa\cdots}_{n}\ \underbrace{bb\cdots}_{n}$

$|a| =$

$|ab| =$

$|x| = |abb| =$

$x.y = xy = |a$

$x = \lambda \quad x.y =$

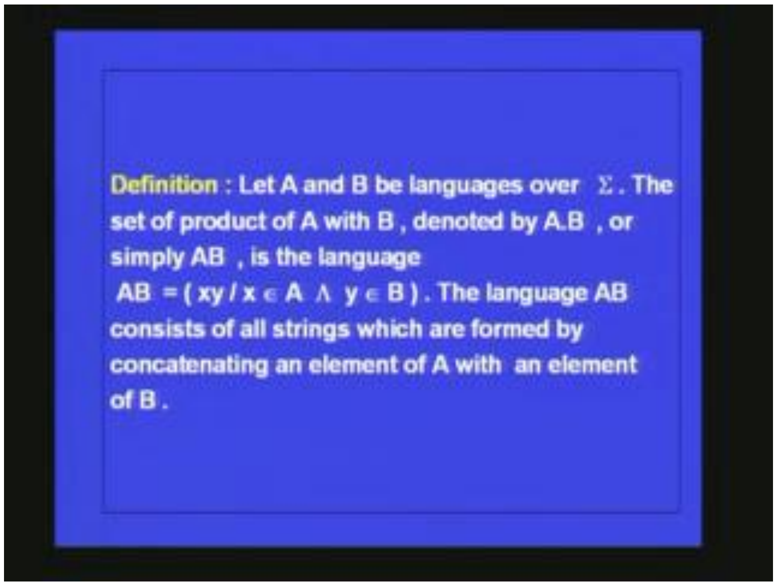
$y = \lambda \quad x.y =$

$|xy| = |x| +$

$|abbcba| = |a$

Now we had defined concatenation of two strings. Next we have to define concatenation of two languages or two sets of strings. How do we define this? Let A and B be languages over sigma, that is A and B are two languages. The set product of A with B or the concatenation of A with B or you can denote it as A dot B or you can say AB is the language AB equal to xy where x belongs to A and y belongs to B. That is AB will consist of strings of the form xy where x will belong to A and y will belong to B. The language AB consists of all strings which are formed by concatenating an element of A with an element of B.

(Refer Slide Time: 14.06)



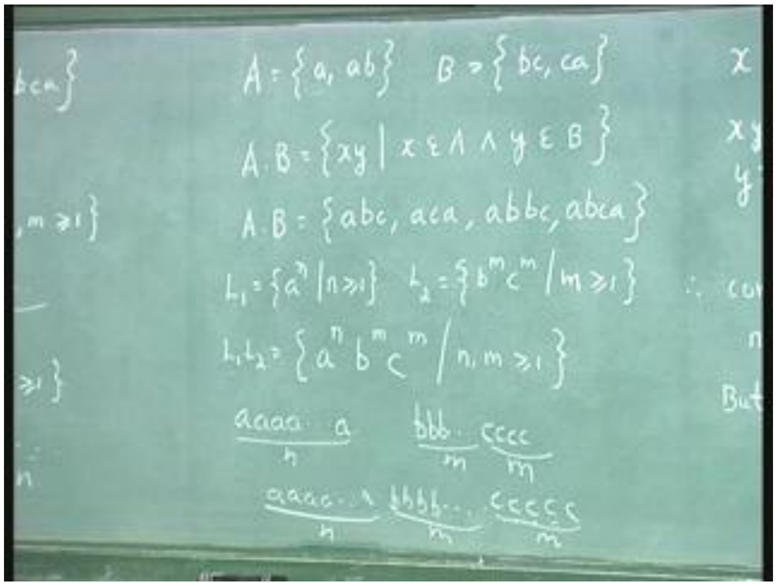
Definition : Let A and B be languages over Σ. The set of product of A with B , denoted by A.B , or simply AB , is the language AB = (xy / x $\in$ A $\wedge$ y $\in$ B). The language AB consists of all strings which are formed by concatenating an element of A with an element of B .

Let L_1 or let me say AB because, we have used A and B in the definition. Let A consists of two strings a, ab and B consists of two string bc, ca. Then the concatenation of A with B, actually by definition it is the set of strings of the form xy where x belongs to A and y belongs to B. Now in this example A dot B will be, a concatenated with bc will give you abc, a concatenated with ca will give you aca, ab concatenated with bc will give you abbc, and ab concatenated with ca will give you abca. So A dot B are concatenation of A with B where A and B are sets of strings.

In this example A has two strings and B has two strings, so AB has four strings, a concatenated with bc, a concatenated with ca, ab concatenated with bc and ab concatenated with ca. Let us also consider an infinite example. Let say L_1 equal to a power n n greater than or equal to 1 and L_2 is equal to b power m, c power m, m greater than or equal to 1. Then what is the concatenation of L_1 and L_2. That will have strings of the form xy where x belongs to L_1 y belongs to L_2 that is it will have strings of the form a power n, b power m, c power m where n and m are greater than or equal to 1.

That is; L_1 has strings of the form aaaa…a and L_2 has strings of the form bbb followed by an equal number of cs. So L_1 and L_2 will have strings of the form aaaa n a's followed by bbb m b's and followed by an equal number of cs where n and m can be any integers greater than are equal to 1. So the language will consist of strings of the form, a string of a's followed by a string of b's with an equal number of cs. The number of b's and c's should be equal. So it is denoted as a power n, b power m, c power m, n and m are greater than or equal to 1.

(Refer Slide Time: 18.38)



$A = \{a, ab\}$ $B = \{bc, ca\}$

$A.B = \{xy \mid x \in A \wedge y \in B\}$

$A.B = \{abc, aca, abbc, abca\}$

$L_1 = \{a^n \mid n \geq 1\}$ $L_2 = \{b^m c^m / m \geq 1\}$

$L_1 L_2 = \{a^n b^m c^m / n, m \geq 1\}$

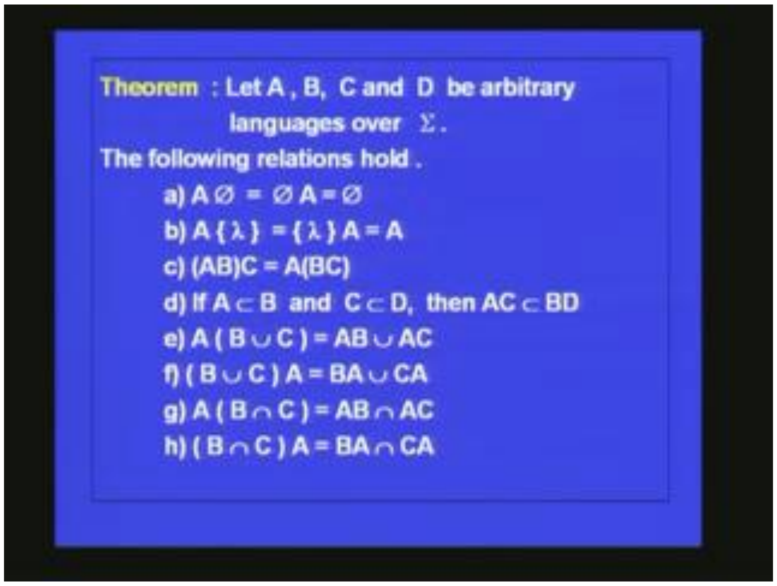
$\underbrace{aaaa \ldots a}_{n}$ $\underbrace{bbb \ldots}_{m}$ $\underbrace{cccc}_{m}$

$\underbrace{aaaa \ldots a}_{n}$ $\underbrace{bbbb \ldots}_{m}$ $\underbrace{ccccc}_{m}$

The concatenation of two languages is defined in this manner. Now we have certain properties of this concatenation over languages. Let us see what they are. Let A B C D be arbitrary languages over sigma. We have taken an alphabet sigma and A B C D are some languages over sigma. Then the following relations hold: The first one says that A phi into phi A equal to phi, that is, when A concatenated with the empty set it consists of strings of the form xy but you cannot have y because this is an empty set, so the result is empty set. A concatenated with lambda or lambda concatenated with A will give you only A because this will have strings of the form xy where y is lambda. So when x into lambda is equal to x so you only get strings from A.

So in a similar manner you can prove that if you concatenate with the set consisting of the empty string only you will get the same set. Note the difference between these two, if you concatenate with the empty set, you get the empty set but if you concatenate with the set consisting of the empty string alone you get the same set. And because of the associative property of concatenation of strings, we can very easily verify that ABC is equal to A into BC. That is this will consist of strings of the form xyz where x belongs to A, y belongs to B, and z belongs to C. And we know that the operation is associative, so you will have ABC is equal to ABC. And also you have, if a language A is contained in B and another language C is contained in D then AC will be contained in BD. This is because AC will consist of strings of the form xy where x belongs to A and y belongs to C.

(Refer Slide Time: 18.44)



Theorem : Let A , B, C and D be arbitrary languages over Σ.

The following relations hold .

a) $A\varnothing = \varnothing A = \varnothing$

b) $A\{\lambda\} = \{\lambda\}A = A$

c) $(AB)C = A(BC)$

d) If $A \subset B$ and $C \subset D$, then $AC \subset BD$

e) $A(B \cup C) = AB \cup AC$

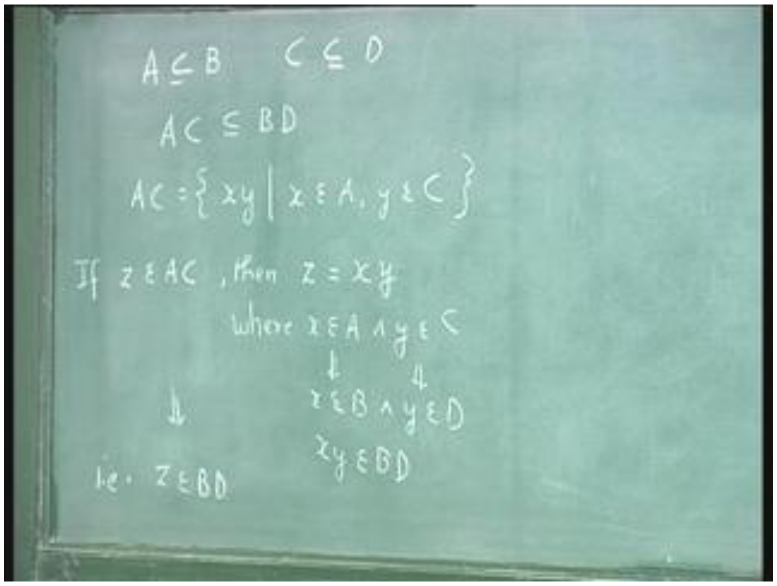
f) $(B \cup C)A = BA \cup CA$

g) $A(B \cap C) = AB \cap AC$

h) $(B \cap C)A = BA \cap CA$

A is contained in B and C is contained in D, so you want to show AC is contained in BD. Now, AC has strings of the form xy where x belongs to A and y belongs to C. So if z belongs to AC then you can write z in the form xy where x belongs to A and y belongs to C. Now A is contained in B, so x belongs to A would imply x belongs to B and y belongs to C, C is contained in D, so this will imply y belongs to D and if x belongs to B and y belongs to D xy will belong to BD that is z belongs to CD. So you conclude that z belongs to AC that will imply z belongs to BD. So that means AC is contained BD.

(Refer Slide Time: 23.30)



$A \subseteq B \quad C \subseteq D$

$AC \subseteq BD$

$AC = \{ xy \mid x \in A, y \in C \}$

If $z \in AC$, then $z = xy$

where $x \in A \wedge y \in C$

$\Downarrow \quad \Downarrow$

$x \in B \wedge y \in D$

$\Downarrow$

$xy \in BD$

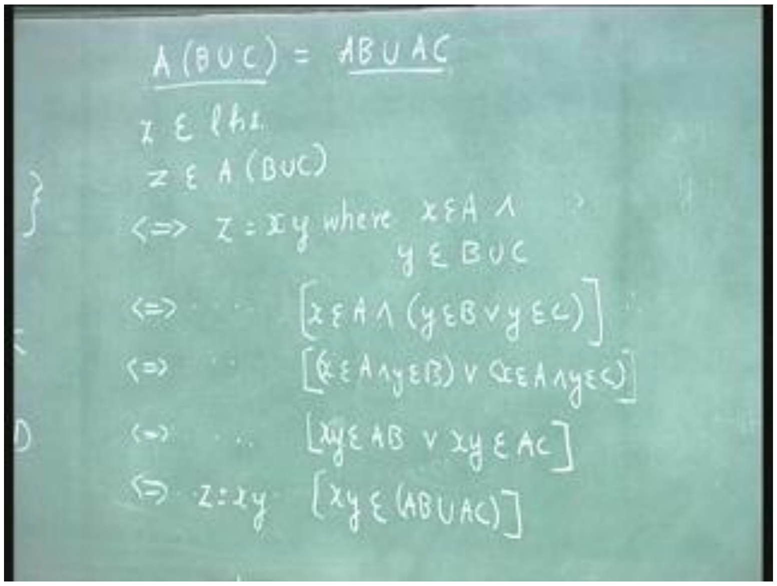
i.e. $z \in BD$

Then concatenation is distributive with respect to union and intersection. These four rules say that. So A concatenated with B union C is equal to AB union AC and B union C concatenated with A gives you BA union CA. A concatenated with B intersection C gives you AB intersection AC and B intersection Cconcatenated with A gives you BA intersection CA. Let us take one and prove then the others are similar.

Take the first one that is A concatenated with B union C gives you AB union AC. That is you want to prove that A B union C is equal to AB union AC. How do you prove that? Now, suppose a string z belongs to lhs that is z belongs to A B union C, this is equivalent to saying z is of the form xy where x belongs to A and y belongs to B union C. That is equivalent to saying z is equal to xy where x belongs to A and y belongs to B union C, this I can write as y belongs to B or y belongs to C. This is equivalent to saying z is of the form xy, where when you distribute this x belongs to A and y belongs to B or x belongs to A and y belongs to C. That is equivalent to saying x belongs to A and y belongs to B would mean x belongs to AB or here xy this is xy belongs to AB and here xy belongs to AC. So this is equivalent to saying xy belongs to AB union AC. So z belonging to A concatenated with B union C is equivalent to saying z is of the form xy and xy belongs to AB union AC so these two are equivalent.

In the similar manner we can prove the other three equalities. This is because concatenation is distributive with respect to union and intersection.

(Refer Slide Time: 27.28)



$\underline{A(B \cup C)} = \underline{AB \cup AC}$

$z \in lhs$

$z \in A(B \cup C)$

$\Leftrightarrow z = xy$ where $x \in A \wedge y \in B \cup C$

$\Leftrightarrow [x \in A \wedge (y \in B \vee y \in C)]$

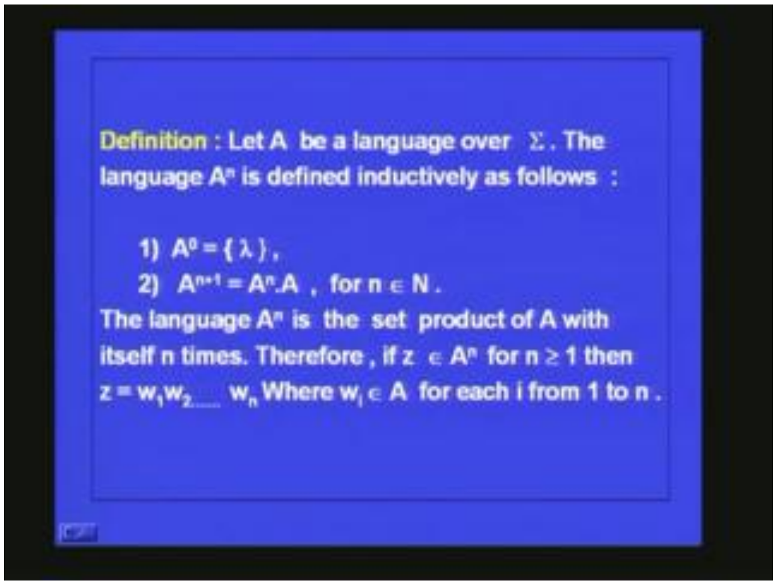
$\Leftrightarrow [(x \in A \wedge y \in B) \vee (x \in A \wedge y \in C)]$

$\Leftrightarrow [xy \in AB \vee xy \in AC]$

$\Leftrightarrow z = xy \ [xy \in (AB \cup AC)]$

Now we have defined what is the power of a string? If x is a string what is x power n, what is x power 0 and so on. If you have a language A, then how do you define the power of a language. We have defined the concatenation of two languages A and B and we have also seen that the concatenation operator when with respect to languages is associative that is ABC is equal to A into BC, because of this with out any problem we can define A power n. Let A be a language over sigma the language A power n is defined inductively as follows: A power 0 is taken to be just the empty string lambda and A power n plus 1 is equal to A power n concatenated with A where n is a natural number. And this we can define because of the associative property with out any difficulty.

(Refer Slide Time: 28.14)



Definition : Let A be a language over Σ. The language A^n is defined inductively as follows :

1) $A^0 = \{ \lambda \}$,
2) $A^{n+1} = A^n.A$, for $n \in N$.

The language A^n is the set product of A with itself n times. Therefore , if $z \in A^n$ for $n \geq 1$ then $z = w_1 w_2 \ldots w_n$ Where $w_i \in A$ for each i from 1 to n .

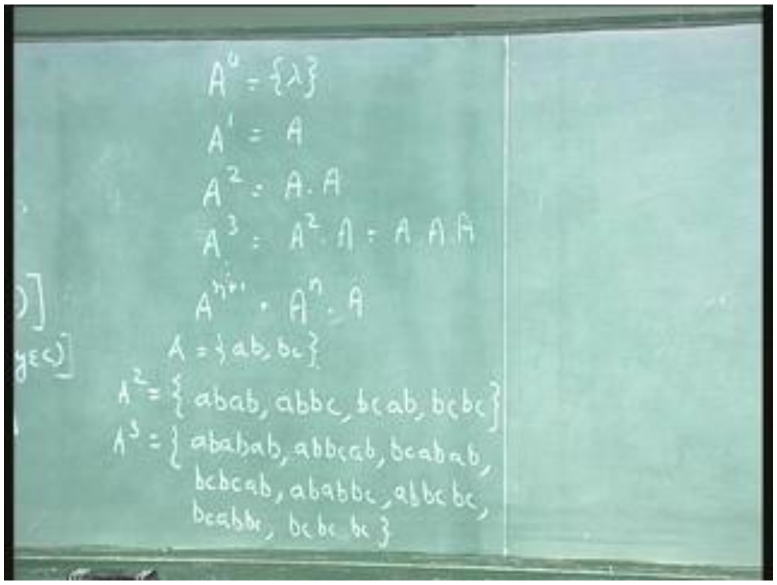
The language A power n is the set product of A with itself n times. Therefore if z belongs to a power n then z can be written in the form w_1 w_2 w_3 w_n where each w_i will belong to A for each i from 1 to n. So how do we define A power n? Some A is a set of strings then A power 0 consists of just one string lambda the empty string, A power 1 is just A itself, and A power 2 is A into A and A cubed is A squared into A, that you can write as A into A into A, that is A concatenated with A concatenated with A without any problem because of the associativity property. Now A power n is n plus 1 is defined as A power n into A.

Let us take an example say A consists of ab bc then A power 0 will be lambda A power 1 will be A that is itself, what will be A square? That will be A concatenated with A and that will have strings of the form abab, abbc, then bcab, and bcbc. So this concatenated itself will give you this, this concatenated with bc gives you this, bc concatenated with ab will give you this, and bc concatenated with bc will give you this. So A squared will have these strings, A cubed will have these strings concatenated with ab and bc.

So how many strings you can expect? You can expect about 8 strings ababab concatenated with ab will give you this, abbc concatenated with ab will give you this, bcab concatenated with ab will give you this then bcbc concatenated with ab will give you this. So you have concatenated each of this with ab. Now you concatenate each of this with bc, so that will give you ababbc, abbcbc then bcabbc, bcbcbc. So these are the string which belongs A cubed. Similarly you can define A power4, A power 5 etc and until A power n.

So in general A power n will have strings of the form x_1 x_2 x_n where each of the xi where i varying from 1 to n will belong to A. So here, A consists of AB and BC. So each of these xi will be AB or BC. This is about the power of language. We shall see some simple properties about this.

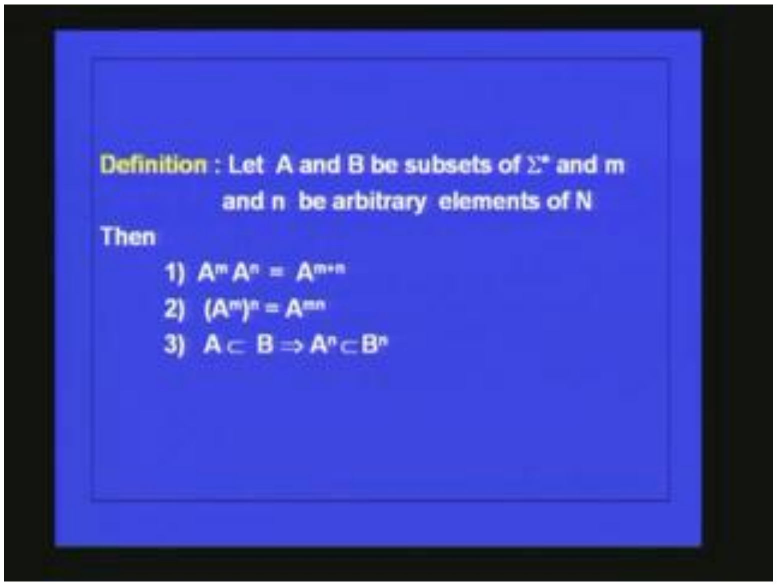
(Refer Slide Time: 32.02)



$$A^0 = \{\lambda\}$$
$$A^1 = A$$
$$A^2 = A.A$$
$$A^3 = A^2.A = A.A.A$$
$$A^{n+1} = A^n.A$$
$$A = \{ab, bc\}$$
$$A^2 = \{abab, abbc, bcab, bcbc\}$$
$$A^3 = \{ababab, abbcab, bcabab, bcbcab, ababbc, abbcbc, bcabbc, bcbcbc\}$$

Let A and B be subsets of sigma star and mn be arbitrary elements of N. That is m and n are some natural numbers. Then you can very easily check this. A power m into A power n will give you A power m plus n. A power m whole power n will give you A power mn and A contain in B will imply A power n will be contain in B power n. You can very easily check these things and they just follow from the definition.

(Refer Slide Time: 32.56)



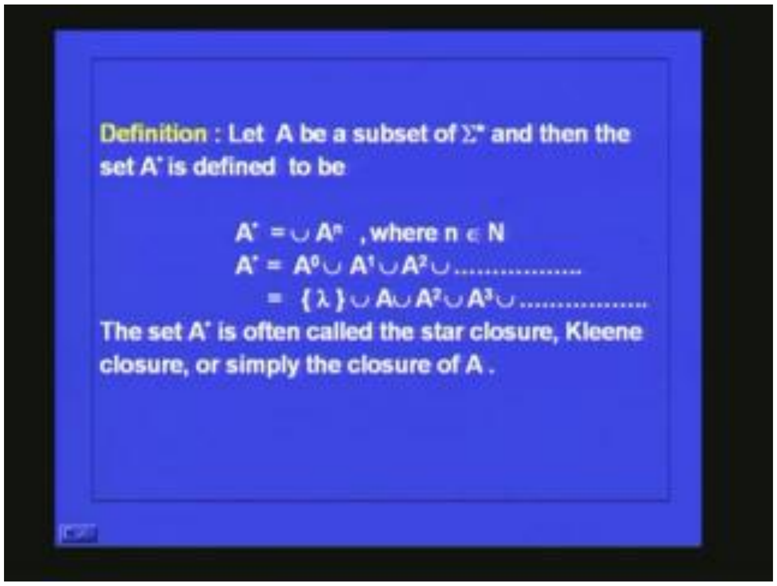
Definition : Let A and B be subsets of Σ^* and m and n be arbitrary elements of N

Then

1) $A^m A^n = A^{m+n}$
2) $(A^m)^n = A^{mn}$
3) $A \subset B \Rightarrow A^n \subset B^n$

Next we define what is mean by a kleene closure of a language. Let A be a subset of sigma star, then A star, the kleene closure is defined to be A star is union A power n where n belongs to N, that is union i is equal to 0 to infinity A power n. That is how you say that A star is A power 0 A power 1 A power 2 A power 3 etc, this is an infinite union. And of course A power 0 you know that it is lambda, so it is lambda A A squared etc. The set A star is often called the star closure or kleene closure or simply the closure of A.

(Refer Slide Time: 33.36)



Definition : Let A be a subset of Σ^* and then the set A^* is defined to be

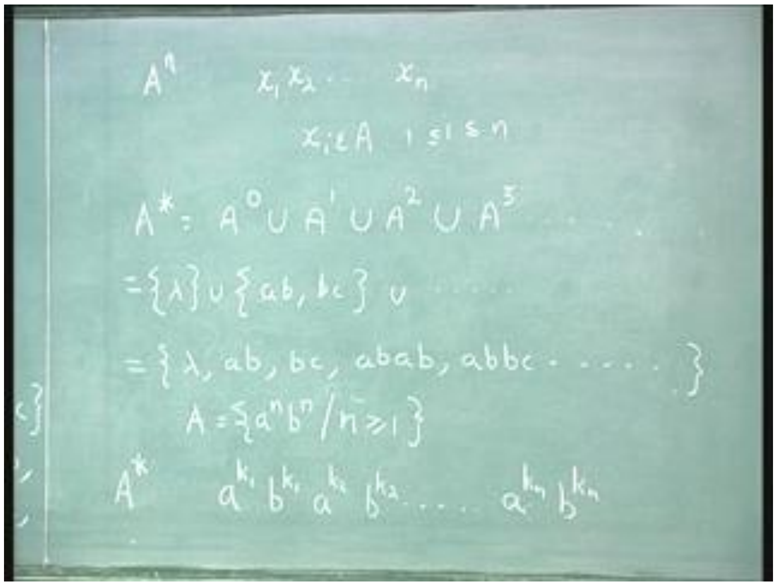
$$A^* = \cup A^n \quad \text{, where } n \in N$$
$$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots\ldots\ldots\ldots\ldots$$
$$= \{\lambda\} \cup A \cup A^2 \cup A^3 \cup \ldots\ldots\ldots\ldots\ldots$$

The set A^* is often called the star closure, Kleene closure, or simply the closure of A .

Now let us take this same example. A is abbc so A power 0 is lambda, A power 1 is A and this A squared is abab, abbc, bcab, bcbc, and so on. Now what is A star here? A star is A power 0 union, it is an infinite union. A star this is called kleene closure of A union A_2 union A_3 and so on. This is an infinite union. And here it will have strings of the form in this particular example lambda union, what does A have ab, bc then what ever strings are there in A squared will also be there. So A star will have strings of the form lambda ab, bc, abab, abbc. We have taken the example where A is a finite set it will hold in a similar manner for infinite sets also.

For example, if we take A to be a power n, b power n, n greater than or equal to 1, then A star will consist of strings of the form a power k_1, b power k_1, a power k_2; b power k_2 etc a power k_n , b power k_n. So it will have some strings of the form a power k_1 b power k_1, a power k_2, b power k_2 and so on where n is greater than or equal to 0 and k_1 k_2 etc are greater than or equal to 1. This will consist of the empty string also and just strings of the form a power n b power n or strings of the form a power n b power n, a power m b power m and so on like that. You will have strings like a power k_1, b power k_1, a power k_2, b power k_2 and so on. So in the kleene closure you can define with respect to a finite language or infinite language in both cases.

(Refer Slide Time: 37.20)



$A^n \quad x_1 x_2 \cdots x_n$

$x_i \in A \quad 1 \le i \le n$

$A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \ldots$

$= \{\lambda\} \cup \{ab, bc\} \cup \ldots$

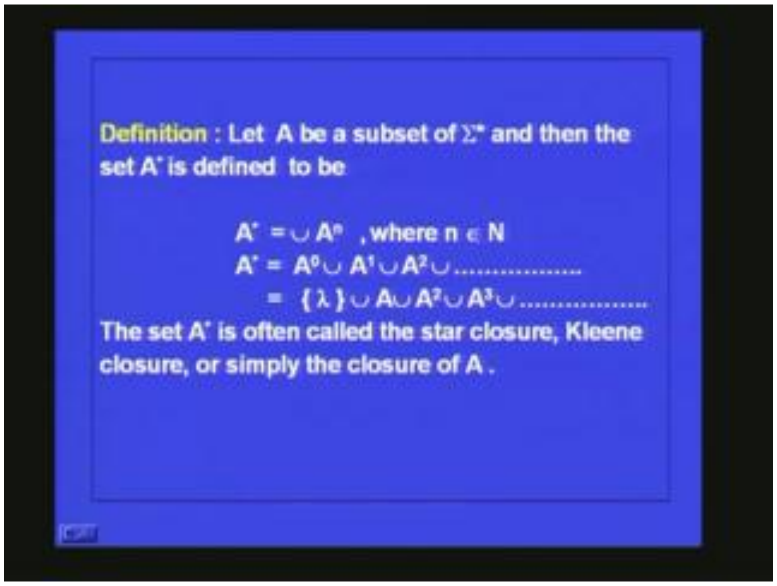
$= \{\lambda, ab, bc, abab, abbc \ldots\}$

$A = \{a^n b^n / n \geq 1\}$

$A^* \quad a^{k_1} b^{k_1} a^{k_2} b^{k_2} \ldots a^{k_n} b^{k_n}$

You also have what is known as A to the power plus or positive kleene closure or epsilon kleene closure. It is defined to be A to the power plus is equal to union A power n where n is greater than or equal to 1. That is here in A plus it starts with A power 1, the infinite union starts with A power 1 and not with A power 0. In the earlier one it started with A power 0, A power 1 etc. Here we start with A power 1, A power 2, A power 3. This is called positive closure or epsilon free kleene closure. The set A to the power plus is often called the positive closure of A.

(Refer Slide Time: 33.26)



Definition : Let A be a subset of Σ^* and then the set A^* is defined to be

$A^* = \cup A^n$, where $n \in N$

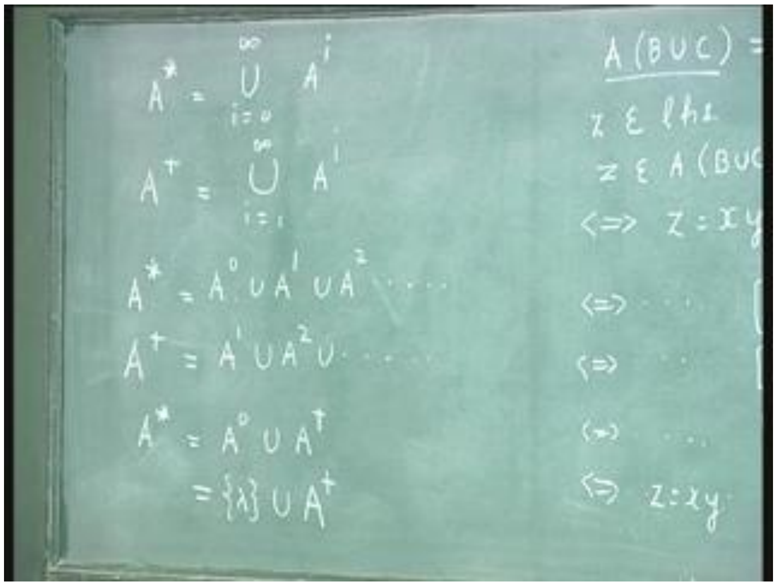
$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots$

$= \{\lambda\} \cup A \cup A^2 \cup A^3 \cup \ldots$

The set A^* is often called the star closure, Kleene closure, or simply the closure of A .

The difference between A to the power plus and A star is this. A star is defined as union i is equal to 0 to infinity A power i and A to the power plus is defined as union i is equal to 1 to infinity A power i where A is a language. So A star is A power 0 union A power 1 union A power 2 etc. A to the power plus is an infinite union, A power 1 union, A power 2 union and so on. So you can very easily see that A star equal to A power 0 union A to the power plus that is equal to lambda because you know that A0 is lambda always Aplus.

(Refer Slide Time: 39.42)



$$A^* = \bigcup_{i=0}^{\infty} A^i$$

$$A^+ = \bigcup_{i=1}^{\infty} A^i$$

$$A^* = A^0 \cup A^1 \cup A^2 \cdots$$

$$A^+ = A^1 \cup A^2 \cup \cdots$$

$$A^* = A^0 \cup A^+$$

$$= \{\lambda\} \cup A^+$$

$A(B \cup C) =$

$z \in lhs$

$z \in A(B \cup C$

$\Leftrightarrow z = xy$

$\Leftrightarrow \cdots$

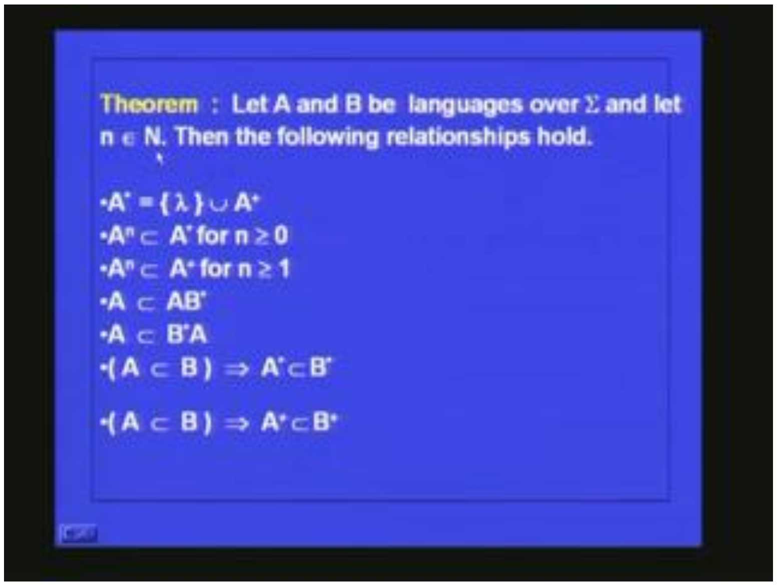
$\Leftrightarrow \cdots$

$\Leftrightarrow \cdots$

$\Leftrightarrow z = xy$

So let A and B be languages over sigma and n is a natural number then the following relationships hold.

(Refer Slide Time 39:45)



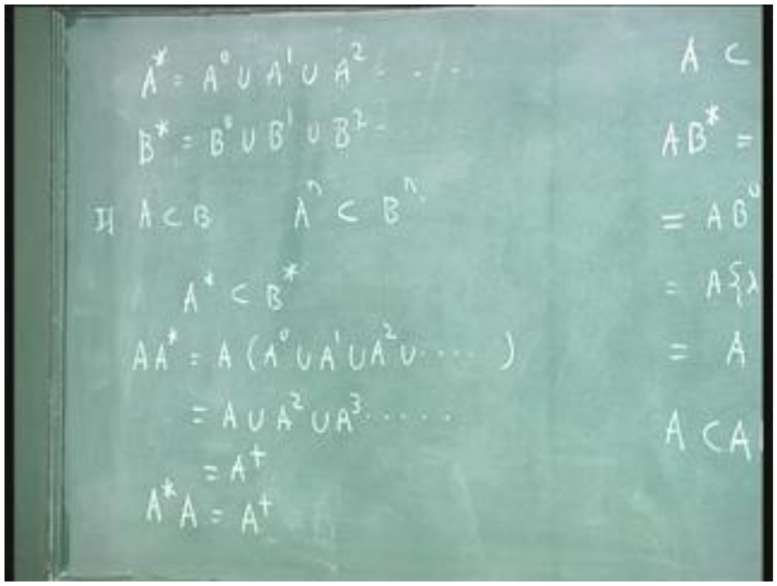
Theorem : Let A and B be languages over Σ and let $n \in N$. Then the following relationships hold.

- $A^* = \{\lambda\} \cup A^+$
- $A^n \subset A^*$ for $n \geq 0$
- $A^n \subset A^+$ for $n \geq 1$
- $A \subset AB^*$
- $A \subset B^*A$
- $(A \subset B) \Rightarrow A^* \subset B^*$
- $(A \subset B) \Rightarrow A^+ \subset B^+$

The first one you have just seen, A star will be equal to lambda union A to the power plus and A power n will be contain in A star for all n greater than or equal to 0 because what is A star? A star is A power 0 union, A power 1 union, A power 2 and so on. So every A power n is part of A star. So A power n is contained in A star for n greater than or equal to 0. Actually this is by definition. In a similar manner you can very easily see that A power n is contained in A to the power plus for n greater than or equal to 1. Then A is contained in AB star. How can you prove that? That is A and B are two languages over sigma, then A is contained in AB star. What is AB star? A concatenated with if you expand this is B power 0 union, B power 1 union, B power 2 and so on. That is AB power 0 union, AB power 1 union and so on. And what is this? This is A into lambda union AB union AB squared union. And A into lambda is A union AB union AB squared and so on. A is part of this, so A is contained in AB star.

Please remember that A into lambda is just A. In a similar manner you can prove that A is contained in B star A and of course A is contained in B will imply A star contained in B star. Why? It is because A is equal to A power 0 union, A power 1 union, A power 2 and this is A star. B star is B power 0 union, B power 1 union. If A is contained in B then A power 0 and B power 0 of course are equal to lambda, A power 1 will be contained in B power 1, A squared will be contained in B squared and each one of this will be contained in this, A power n will be contained in B power n. So with the result A star will be contained in B star.

(Refer Slide Time: 45.06)



$A^* = A^0 \cup A^1 \cup A^2 \cdots$

$B^* = B^0 \cup B^1 \cup B^2 \cdots$

If $A \subset B$ $\quad A^n \subset B^n$

$A^* \subset B^*$

$AA^* = A(A^0 \cup A^1 \cup A^2 \cup \cdots)$

$= A \cup A^2 \cup A^3 \cdots$

$= A^+$

$A^*A = A^+$

$A \subset$

$AB^* =$

$= AB^0$

$= A\{\lambda$

$= A$

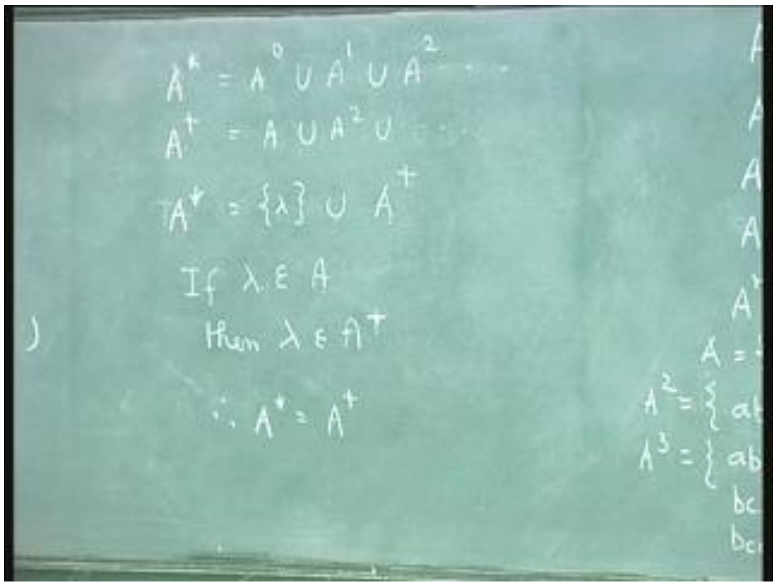
$A \subset A$

The similar result holds when you consider plus also. A contains in B will imply A to the power plus is contained in Bplus. A A star is equal to A star A is equal to Aplus. This you can very easily see like this. A A star will be A concatenated with A power 0, union A power 1, union A power 2 and so on. That will give you A union A squared union A cubed.

Similarly, this will be nothing but Aplus. In a similar manner you can also prove that A star A is also equal to Aplus. If lambda belongs to A then A to the power plusand A star are equivalent.

Why A star is A power 0 union, A power 1 union, A power 2 union and so on. A to the power plus is A union A squared union and so on. So what can you say about A star? A star is A power 0 is lambda union Aplus. This we have already seen. Now if lambda belongs to A, A has lambda so A to the power plus also has lambda. When this already has got lambda finding the union with lambda again the start give anything different it gives the same thing. So if lambda belongs to A then lambda belongs to Aplus. In that case adding this additionally does not make any difference. Therefore A star will be equal to Aplus.

(Refer Slide Time: 46.44)



$A^* = A^0 \cup A^1 \cup A^2 \cdots$

$A^+ = A \cup A^2 \cup \cdots$

$A^* = \{\lambda\} \cup A^+$

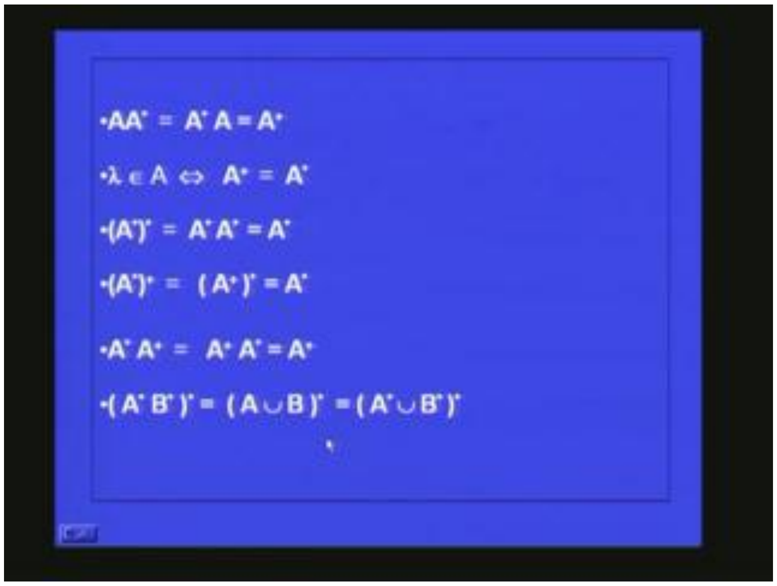
If $\lambda \in A$

then $\lambda \in A^+$

$\therefore A^* = A^+$

These are some of the results again based on the definition of A star and Aplus. A star star is equal to A star A star equal to A star, A star plus will be A to the power plus star equal to A star, A star A to the power plus will be A to the power plus equal to A to the power plus A star and A star B star star will be A union B star A star union B star star. You can verify these things very easily.

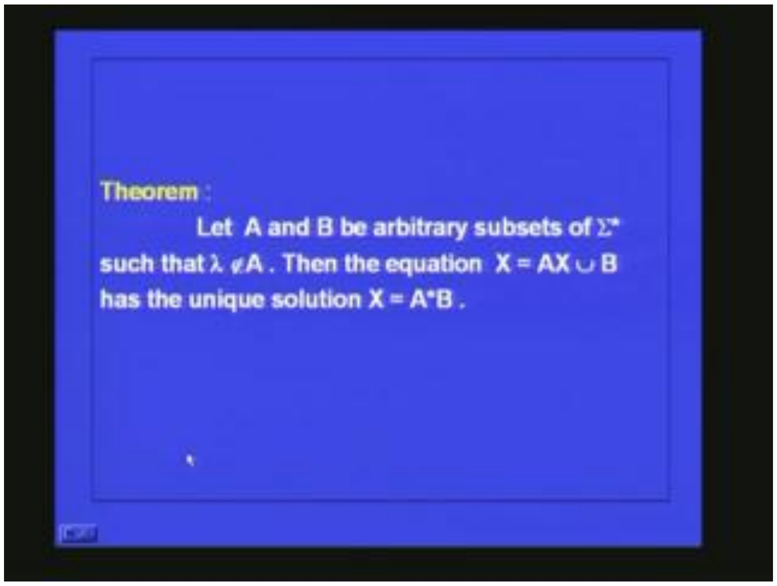
(Refer Slide Time: 43.54)



- $AA^* = A^*A = A^+$
- $\lambda \in A \Leftrightarrow A^+ = A^*$
- $(A^*)^* = A^*A^* = A^*$
- $(A^*)^+ = (A^+)^* = A^*$
- $A^*A^+ = A^+A^* = A^+$
- $(A^*B^*)^* = (A \cup B)^* = (A^* \cup B^*)^*$

There is a small result about this which is very useful in automata theory. Later on this set of result will be used while finding out the regular expression from a deterministic final state automata.

(Refer Slide Time: 47.22)



Theorem :

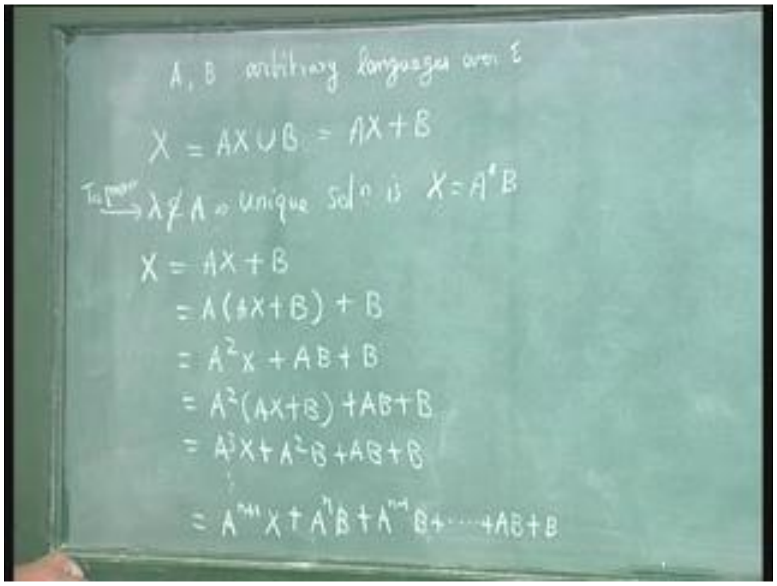
Let A and B be arbitrary subsets of Σ^* such that $\lambda \notin A$. Then the equation $X = AX \cup B$ has the unique solution $X = A^*B$.

Let us consider this theorem. Let A and B be arbitrary subsets of sigma star such that lambda does not belong to A. Then the equation X is equal to AX union B has the unique solution X is equal to A star B. This is a very useful lemma and let us try to prove this theorem. A and B are arbitrary languages over sigma, it is a subset of strings. Then you have an equation X is equal to AX union B. I shall write it as AX plus B where plus is denoting the union. Then this has a unique self.

What you want to show that this has a unique solution X is equal to A star B. And the condition is lambda does not belong to A. Then the unique solution is X is equal to A star B. Let us try to prove this result. This is what you want to prove. So let us write like this: X is equal to AX plus B.

Now again I can replace this X by AX plus B. So that is AX plus B plus B that is again A squared X plus AB plus B that is equal to A squared. Again I replace x by AX plus B plus AB plus B and this will give you A cubed X plus A squared B plus AB plus B and proceeding in this manner you can get A power n plus 1 X plus A power n X plus A power B plus and AB plus B.

(Refer Slide Time: 51.10)



A, B arbitrary languages over Σ

$X = AX \cup B = AX + B$

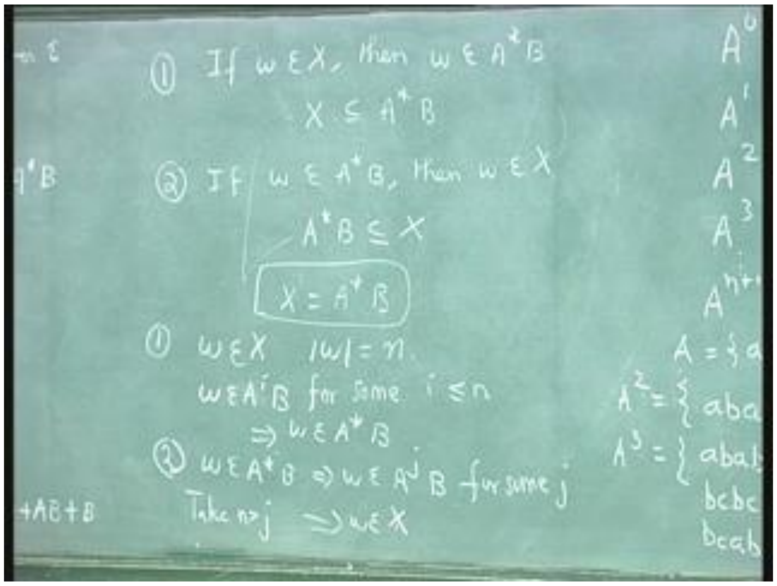
$\xrightarrow{\text{To prove}} \lambda \notin A \Rightarrow$ unique sol^n is $X = A^*B$

$$
\begin{aligned}
X &= AX + B \\
&= A(AX+B) + B \\
&= A^2X + AB + B \\
&= A^2(AX+B) + AB + B \\
&= A^3X + A^2B + AB + B \\
&= A^{n+1}X + A^nB + A^{n-1}B + \cdots + AB + B
\end{aligned}
$$

Now you want to show that X is equal to A star B. For that what you have to show is, if some w belongs to X then w belongs to A star B. You have to prove it into two parts, first part is this if w belongs to x then w belongs to A star B this means X is contained in A star B. The second part is if w belongs to A star B then w belongs to X. This would mean A star B is contained in X. From these two you get X is equal to A star B.

Now let us take the first one. Let w belong to X and the length of w be n. Take a string w belonging to X and a length of the string is n. Now since lambda does not belong to A, any string here the length of any string here will be at least n plus 1. So when you say w is of length n it should belong to this portion. So w will belong to this portion that means w will be A power iB. That means w belongs to A power iB for some i less than or equal to n. That means w belongs to A star B. The second portion is, if w belongs to A star B then w belongs to X. w belongs to A star B implies w belongs to some A power jB for some j. Take n greater than j, then we know that X can be written in this form and w belongs to A power j, so it belongs to this portion, X can be written in this form. X consists of this portion as well as this portion. So it belongs to this portion and so it belongs to X. So this will imply w belongs to X. And so X is a equivalent to A star B. This is a unique solution.

(Refer Slide Time: 54.19)



① If $w \in X$, then $w \in A^*B$

$X \subseteq A^*B$

② If $w \in A^*B$, then $w \in X$

$A^*B \subseteq X$

$\boxed{X = A^*B}$

① $w \in X \quad |w| = n.$

$w \in A^iB$ for some $i \leq n$

$\Rightarrow w \in A^*B$

② $w \in A^*B \Rightarrow w \in A^jB$ for some j

Take $n > j \Rightarrow w \in X$

A^0 A^1 A^2 A^3 A^{n+1}

$A = \{a$

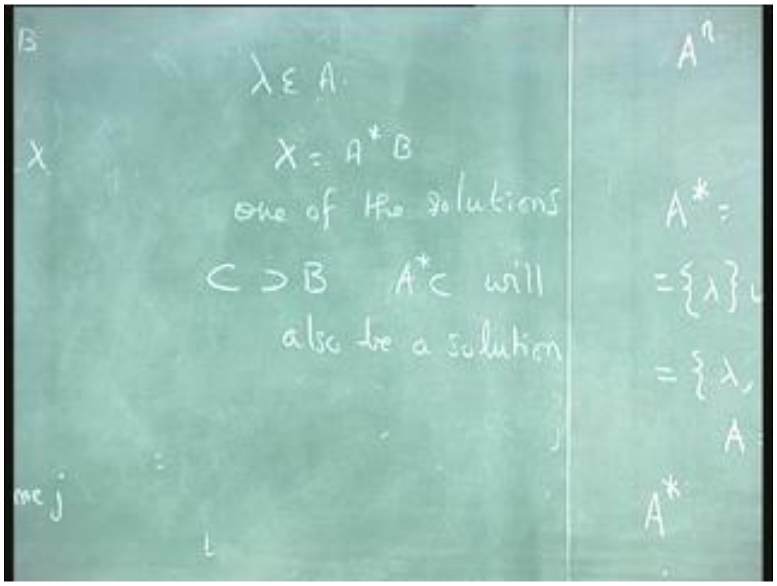
$A^2 = \{aba$

$A^3 = \{abab$, bcbc, bcab

A^*B $+AB+B$

Now we are able to prove this because lambda does not contain A and that is why we are able to say that for this portion the minimum length of string should be at least n plus 1. Now, if that condition is dropped, that is if you have lambda belongs to A then what happens? In this case again X is equal to A star will be one of the solutions. The solution will not be unique. Any C containing B or any C containing B A star C will also be a solution. This particular lemma is very useful when you want to find a regular expression from a deterministic final state automata.

(Refer Slide Time: 55.19)



$\lambda \in A.$

$X = A^*B$

one of the solutions

$C \supset B \quad A^*C$ will also be a solution

A^n

$A^* =$

$= \{\lambda\} \cup$

$= \{\lambda,$

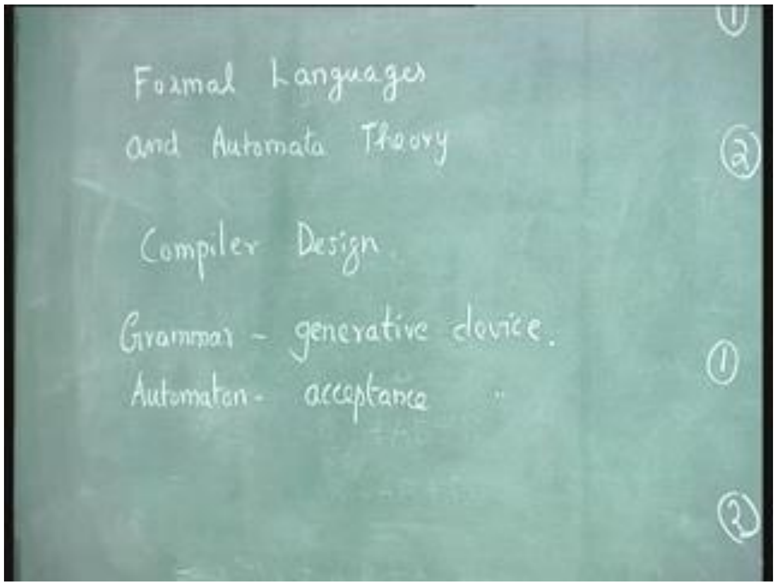
A

A^*

We shall learn about final state automata later, that is towards the end of this course.

Now this study about set of strings is very useful in formal languages and automata theory. And that finds application in compiler design. This study is useful in formal languages and automata theory. This study itself finds use in compiler design because in any program, for example take a Pascal program, it is taken as a sequence of symbols and what are syntactically correct programs and what are not syntactically correct programs is defined by means of a grammar which defines that programming languages. For each programming language like Pascal, earlier languages like algal sixty, Fortran you consider a grammar which will generate all syntactically correct programs in that particular language. The study of formal language and automata theory was really motivated by the use of it in compiler design.

(Refer Slide Time: 57.46)



Formal Languages
and Automata Theory

Compiler Design

Grammar - generative device.
Automaton - acceptance "

So you talk about a grammar which is a generative device and you also talk about an automaton which is an acceptance device. Thus the study about strings and operations on sets of strings are very useful in formal languages in automata theory which itself very useful in compiler design theory.