Theory of Computation

Prof. Kamala Krithivasan

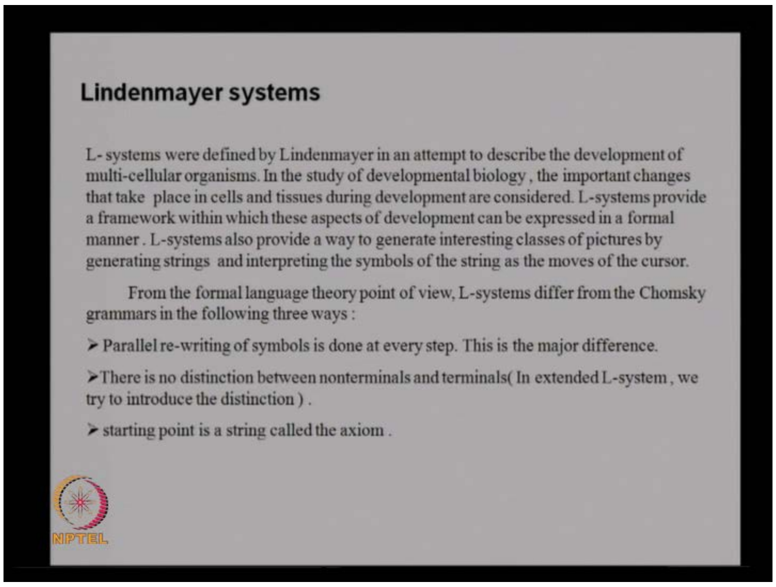
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 39

L-Systems

(Refer Slide Time: 00:17)



Lindenmayer systems

L- systems were defined by Lindenmayer in an attempt to describe the development of multi-cellular organisms. In the study of developmental biology , the important changes that take place in cells and tissues during development are considered. L-systems provide a framework within which these aspects of development can be expressed in a formal manner . L-systems also provide a way to generate interesting classes of pictures by generating strings and interpreting the symbols of the string as the moves of the cursor.

From the formal language theory point of view, L-systems differ from the Chomsky grammars in the following three ways :

- Parallel re-writing of symbols is done at every step. This is the major difference.
- There is no distinction between nonterminals and terminals(In extended L-system , we try to introduce the distinction) .
- starting point is a string called the axiom .

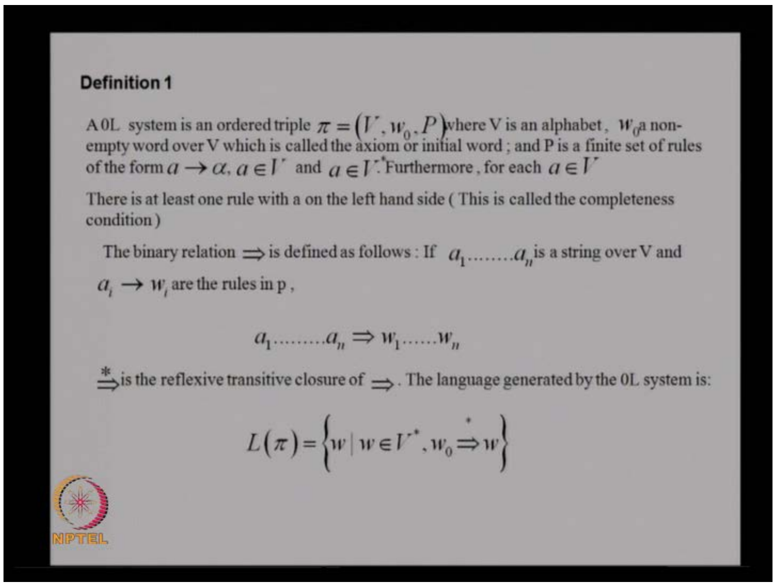
Today, we shall consider one more topic namely Lindenmayer systems. These are very useful in creating pictures, and in computer imagery. So, what are these systems? Lindenmayer systems were defined in 1968 by professor Lindenmayer, who is theoretical biologist; he was looking into the growth of organism plants and other things. And how they could be modeled using a formal system. The formal language theory with the chomskyan hierarchy was present at the time. And he changed the model and created a new model, which is parallel in rewriting. And it could explain the behavior of the growth of organisms. So, these are defined L-systems, because lindenmayer defined the them they are called L-systems they were defined in nineteen sixty eight.

And in developmental biology, the important changes that take place in the cells, and the during the development can be modeled by this systems. They provide a frame work within which these aspects of development can be expressed in a formal manner. Now, the formal language theory point of view, L-systems differ from Chomskyer hierarchy in three ways. In Chomskyan hierarchy we saw that the rewriting is sequential. At any step only one symbol or one substring is rewritten as something else, the left hand side is replaced by the right hand side of a rule. And only one rule is used at a particular step.

But here in Lindenmayer system, we consider parallel rewriting of symbols in every step, this is a major difference. And then we do not make any distinction between non terminals and terminals here. The reason for this is when you try to model the growth of an organism there is nothing like, non terminal and terminal all symbol are equivalent. Whereas, from the parsing point of view when you consider, a natural language sentence you have syntactic categories. And the word generated by the language so there is naturally a distinction between non terminals and terminals. So, here there is no purpose in having a non terminals, and terminals if you want to model the growth of organisms.

But from formal language theory point of view, sometimes it is advantages to have the distinction and. So, later on we defined extended L-systems where, we make a distinction between these two categories. Not only that in Chomskyan hierarchy, we have always seen that the start symbol was a single symbol s. Even when we use regulate rewriting the start symbol was only, a single symbol s which is noted as start symbol or the sentence symbol. Now, here the starting point will be a string, that is called the axiom. So, with these distinctions let us see how the system is defined.

(Refer Slide Time: 03:45)



Definition 1

A 0L system is an ordered triple $\pi = (V, w_0, P)$ where V is an alphabet, w_0 a non-empty word over V which is called the axiom or initial word ; and P is a finite set of rules of the form $a \rightarrow \alpha$, $a \in V$ and $\alpha \in V^*$. Furthermore, for each $a \in V$

There is at least one rule with a on the left hand side (This is called the completeness condition)

The binary relation $\Rightarrow$ is defined as follows : If $a_1 \ldots\ldots a_n$ is a string over V and $a_i \rightarrow w_i$ are the rules in p ,

$$a_1 \ldots\ldots a_n \Rightarrow w_1 \ldots\ldots w_n$$

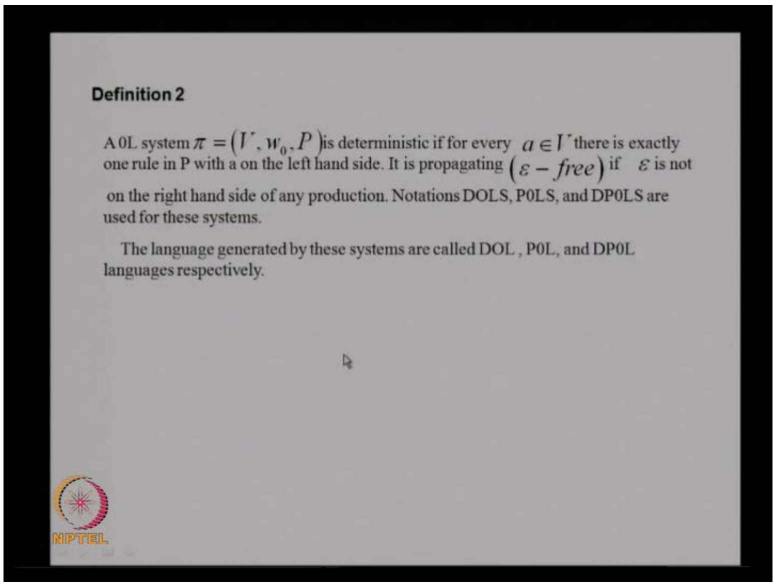
$\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$. The language generated by the 0L system is:

$$L(\pi) = \left\{ w \mid w \in V^*, w_0 \overset{*}{\Rightarrow} w \right\}$$

A zero L-systems, 0 denotes without interaction. A 0 L it is not 0 L it is 0 L. 0 L system is an ordered triple, V w naught P where V is the alphabet. We are considering only one alphabet, and w naught is a string it is a non empty word over V. And that is called the axiom of the system and p is a set of production rules. If they are without interactions the rules are of this form, left hand side you have a symbol you have a symbol from V and on the right hand side you have a string.

So, a belongs to b and alpha belongs to V star now, for each a there should be at least one rule. With a on the left hand side this is called the completeness condition. Because we want to explain the growth for each and every cell of the organism. So, a cell can be represented as a symbol. And every symbol grows or changes and that is denoted by these rules. Now, the relation is defined like this, the derivation step is defined if a 1 a two a n is a string over v and the rules are given by a. I goes to w I, then if you have a one a two a n in one step.

(Refer Slide Time: 05:12)



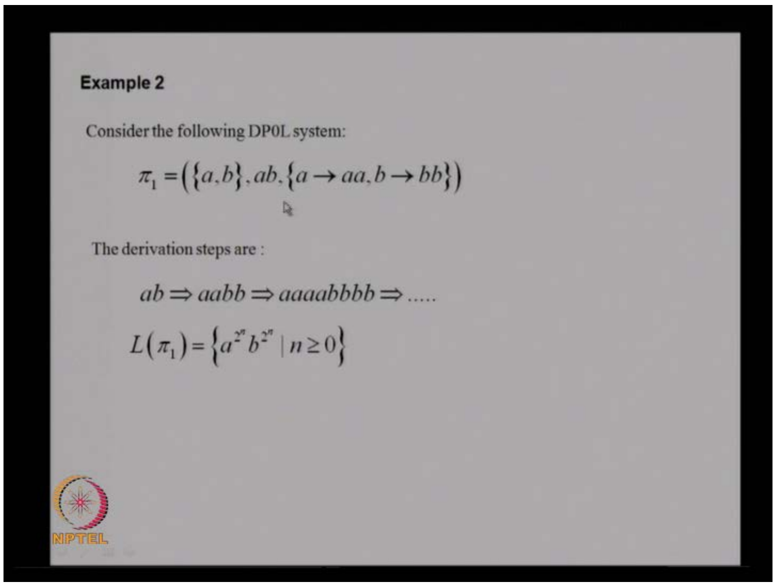
Definition 2

A 0L system $\pi = (V, w_0, P)$ is deterministic if for every $a \in V$ there is exactly one rule in P with a on the left hand side. It is propagating $(\varepsilon - free)$ if ε is not on the right hand side of any production. Notations DOLS, P0LS, and DP0LS are used for these systems.

The language generated by these systems are called DOL , P0L, and DP0L languages respectively.

In the next step all symbols are rewritten simultaneously a 1, is rewritten as w 1 a two is rewritten as w two a three is rewritten as w three a n is rewritten as w n. All steps take place simultaneously all rewriting takes place simultaneously and this is called one step of the derivation. Now, as usual we denote by double arrow star the reflexive transitive closure of w. The language generated by the system is defined as the set of strings, it can be derived from the axiom. w belongs to V star and from the axiom it should be possible to derive w. Take for example, a system like this.

(Refer Slide Time: 06:02)



Example 2

Consider the following DP0L system:

$$\pi_1 = (\{a,b\}, ab, \{a \to aa, b \to bb\})$$

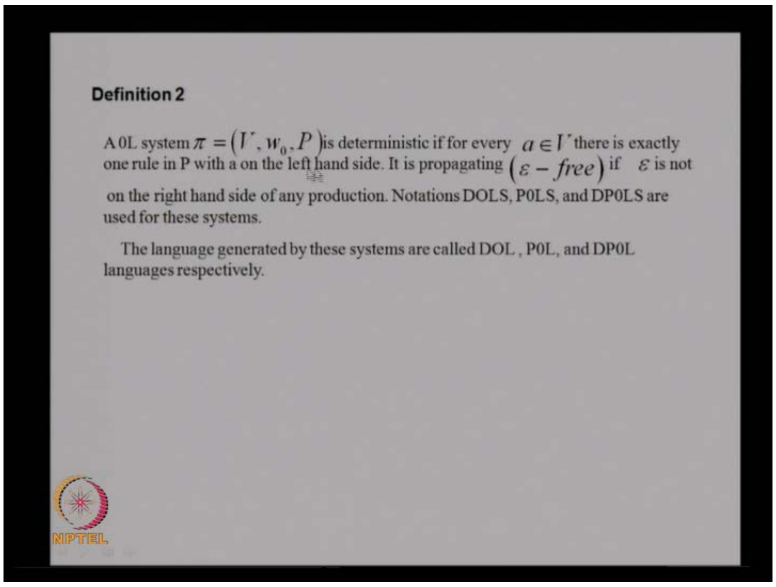
The derivation steps are :

$$ab \Rightarrow aabb \Rightarrow aaaabbbb \Rightarrow .....$$

$$L(\pi_1) = \{a^{2^n} b^{2^n} \mid n \geq 0\}$$

You start with the axiom a b, rule for a is a goes to a a, rule for b is b goes to b b, then starting with a b you have a rewritten as a a, b rewritten as b b. And the next step each a is rewritten as a a, each b is rewritten as b b. So, in two steps you get this and generally the string, generated will be of the form a power 2 power n b power 2 power n. The number of as is doubled in the next step the number of bs is also doubled in the next step,

(Refer Slide Time: 06:43)



Definition 2

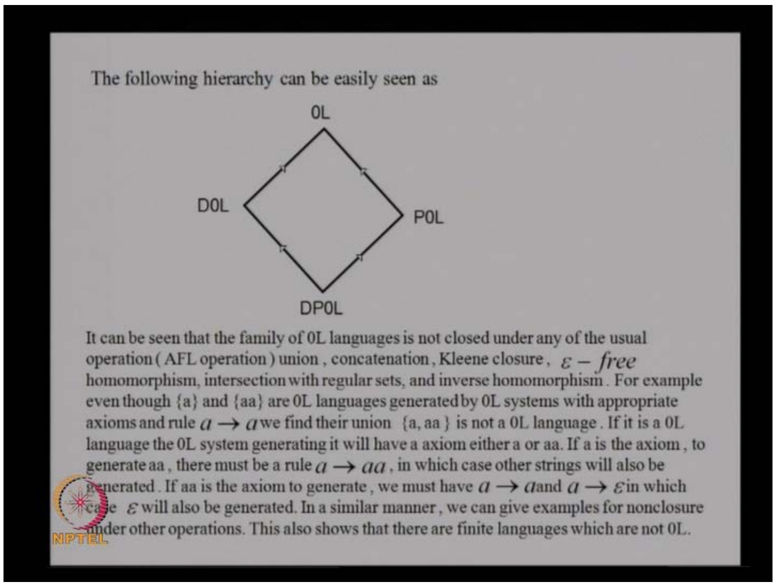
A 0L system $\pi = (V, w_0, P)$ is deterministic if for every $a \in V$ there is exactly one rule in P with a on the left hand side. It is propagating $(\varepsilon - free)$ if ε is not on the right hand side of any production. Notations DOLS, P0LS, and DP0LS are used for these systems.

The language generated by these systems are called DOL , P0L, and DP0L languages respectively.

now, on the left hand side you have a single symbol and, on the right hand side you have a string.

The string can be epsilon free and, if it is epsilon free the system is called a propagating. Now, if you have only one rule for each a then the system is set to be deterministic. A 0 L, system pi which is equal to V W naught P is deterministic, if for every a belonging to V there is exactly one rule in P. And it is propagating if it is epsilon free that is epsilon is not on the right hand side of any production. So, you have a D zero L system you have a p zero L system you have a D P L, zero system which is a combination of deterministic and propagating.

(Refer Slide Time: 07:40)



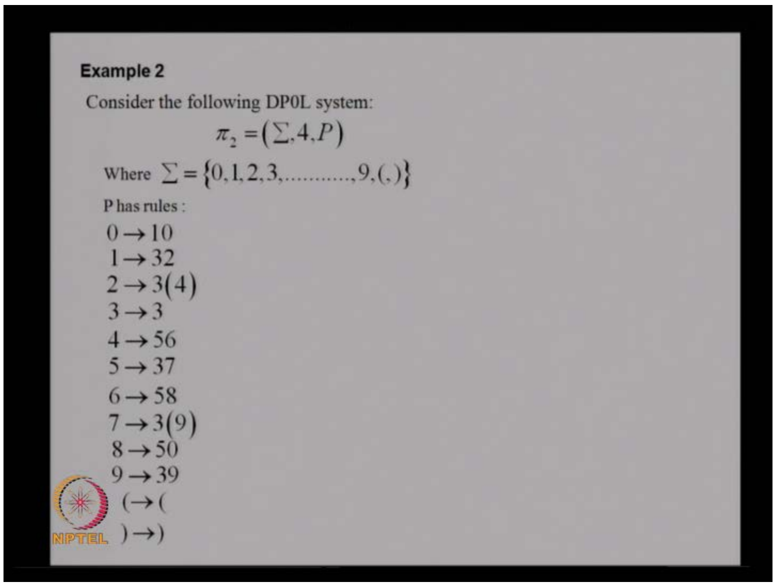
The following hierarchy can be easily seen as

$$\begin{tikzcd} & 0L & \\ D0L \arrow[ur] & & P0L \arrow[ul] \\ & DP0L \arrow[ul] \arrow[ur] & \end{tikzcd}$$

It can be seen that the family of 0L languages is not closed under any of the usual operation (AFL operation) union , concatenation , Kleene closure , $\varepsilon - free$ homomorphism, intersection with regular sets, and inverse homomorphism . For example even though {a} and {aa} are 0L languages generated by 0L systems with appropriate axioms and rule $a \rightarrow a$ we find their union {a, aa } is not a 0L language . If it is a 0L language the 0L system generating it will have a axiom either a or aa. If a is the axiom , to generate aa , there must be a rule $a \rightarrow aa$, in which case other strings will also be generated . If aa is the axiom to generate , we must have $a \rightarrow a$ and $a \rightarrow \varepsilon$ in which case ε will also be generated. In a similar manner , we can give examples for nonclosure under other operations. This also shows that there are finite languages which are not 0L.

The language classes will be like this, if you have DP0L system. It will be contained in 0 L system D0L and that is contained in the general 0 L. Similarly, DP0L will be contained in P0L and that will be contained in the 0 L.

(Refer Slide Time: 07:59)



Example 2

Consider the following DP0L system:

$$\pi_2 = (\Sigma, 4, P)$$

Where $\Sigma = \{0, 1, 2, 3, \ldots\ldots\ldots, 9, (,)\}$

P has rules :

$0 \rightarrow 10$

$1 \rightarrow 32$

$2 \rightarrow 3(4)$

$3 \rightarrow 3$

$4 \rightarrow 56$

$5 \rightarrow 37$

$6 \rightarrow 58$

$7 \rightarrow 3(9)$

$8 \rightarrow 50$

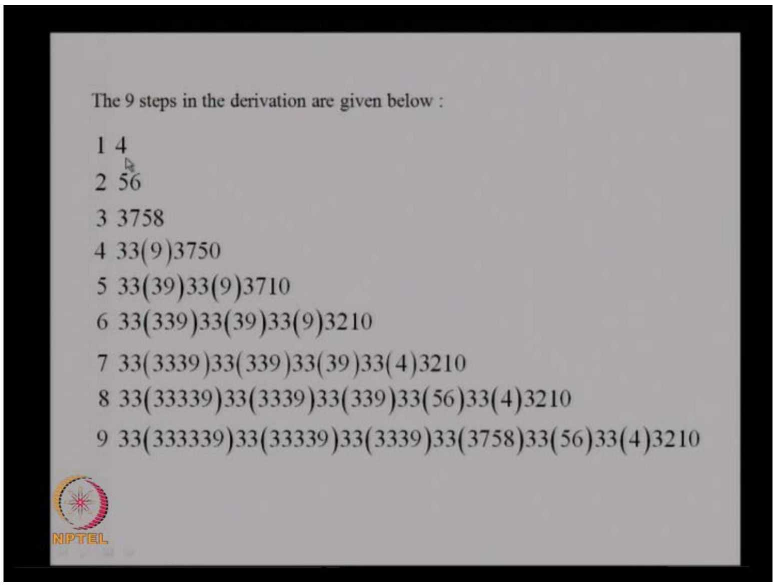
$9 \rightarrow 39$

$(\rightarrow ($

$) \rightarrow)$

Consider one more deterministic propagating 0 L system. Where the alphabet consist of 0,1, 2, 3, 9, left parenthesis right parenthesis. One of the symbols 4 is the axiom, the set of production are given by this 0 goes to 1 0 1 goes to 3 2 2 goes to 3 within bracket 4 and so on these are the rules.

(Refer Slide Time: 08:30)



The 9 steps in the derivation are given below :

1 4

2 56

3 3758

4 33(9)3750

5 33(39)33(9)3710

6 33(339)33(39)33(9)3210

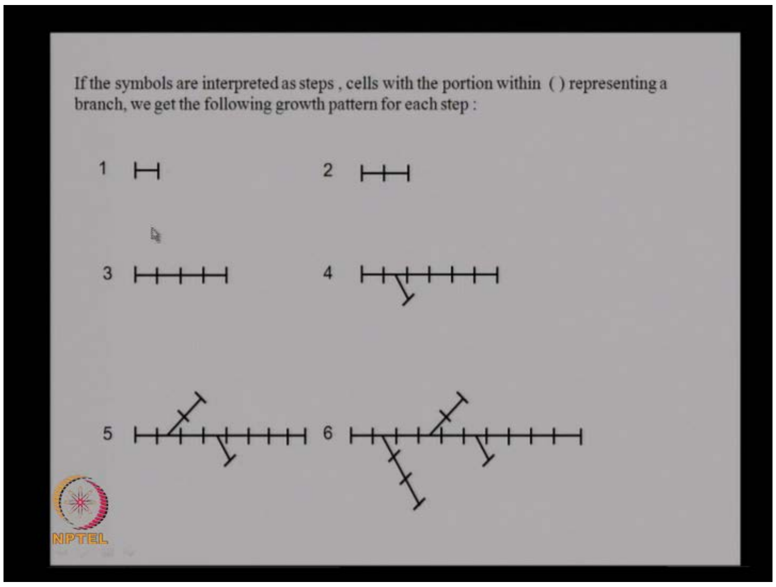
7 33(3339)33(339)33(39)33(4)3210

8 33(33339)33(3339)33(339)33(56)33(4)3210

9 33(333339)33(33339)33(3339)33(3758)33(56)33(4)3210

And with this in the first step you have 4 second step you get 5 6, in the third step you ge 3 7 5 8 and so on. So, proceeding like this in 9 steps you get derivations of this form.Because look at the rules, 0 goes to 1 0 and so on 4 goes to 5 6 and 5 goes to 3 7, 6 goes to 5 8. So, you see 4 goes to 5 6 second step you get 5 6 then 5 goes to 6 7 6 goes to 5 8. In the third step you get this and if you proceed you will get strings, of this form. Actually if you denote the integers, to represent sort stream length. And brackets within brackets you have the branches.

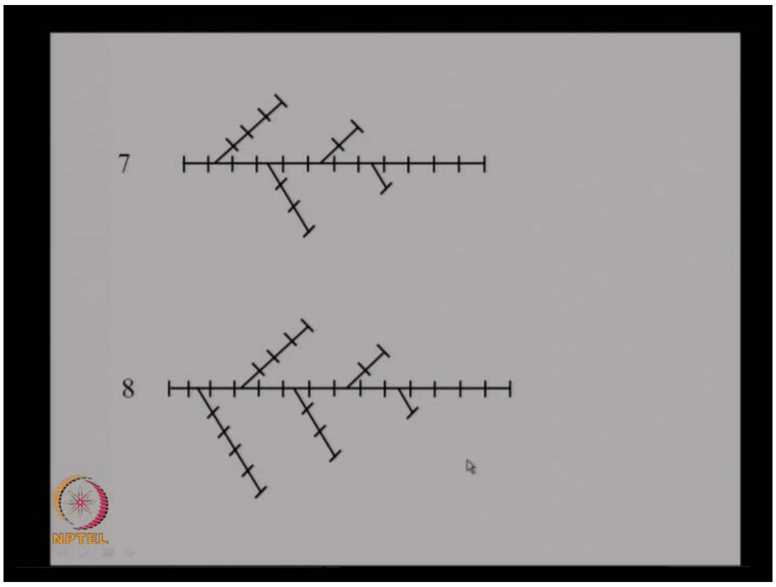
(Refer Slide Time: 09:17)



If the symbols are interpreted as steps , cells with the portion within ( ) representing a branch, we get the following growth pattern for each step :

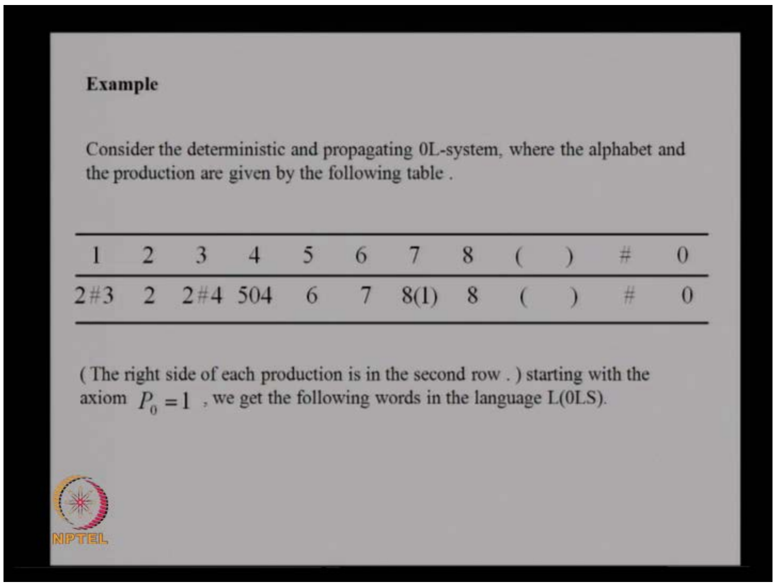
Then you have 1 step 2 step 3 step and so on 4 you have a single branch, then two branches you get, in the next step got three branches and so on.

(Refer Slide Time: 09:26)



So, you find that the steps can be defined by, the growth of a step and branches are developing from this step. And this way the growth of a plant can be described, this is it is with this a that lindenmayer defined the system.

(Refer Slide Time: 09:49)



Example

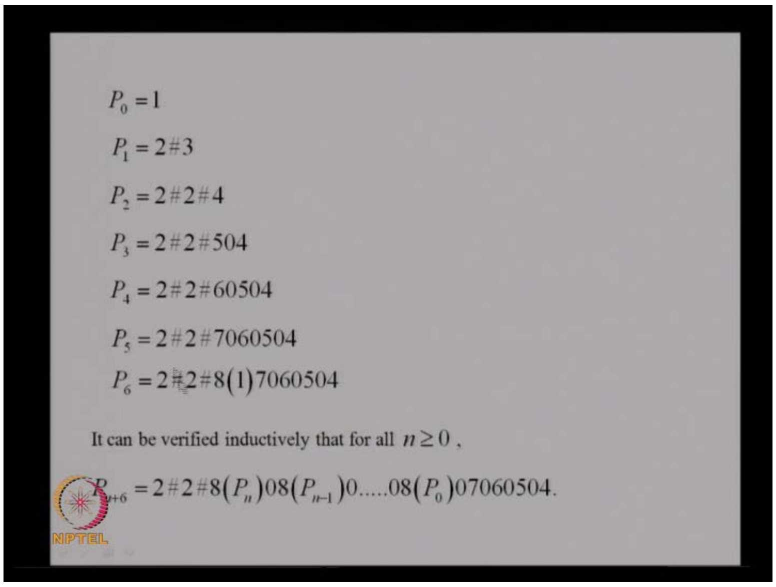
Consider the deterministic and propagating 0L-system, where the alphabet and the production are given by the following table .

1	2	3	4	5	6	7	8	(	)	#	0
2#3	2	2#4	504	6	7	8(1)	8	(	)	#	0

(The right side of each production is in the second row .) starting with the axiom $P_0 = 1$, we get the following words in the language L(0LS).

One more example, we shall consider consider the deterministic and the propagating 0 L system, that is both deterministic and propagating. Because you see that for every, there is, only one rule the symbols are 0 to 1 to 8 0 to 8 0 is here. And three more symbols left parenthesis, right parenthesis hash symbols. The rules are 1 goes to 2 hash 3 2 goes to 2 3 goes to 2 hash 4. 4 goes to 5 0 4, 5 goes to 6, 6 goes to 7 and so on. And the right hand side of the, production is given in the lower row. Left hand side is given in the upper row, and you start with p naught is equal to one the axiom is one if you use this.

(Refer Slide Time: 10:49)


$P_0 = 1$

$P_1 = 2\#3$

$P_2 = 2\#2\#4$

$P_3 = 2\#2\#504$

$P_4 = 2\#2\#60504$

$P_5 = 2\#2\#7060504$

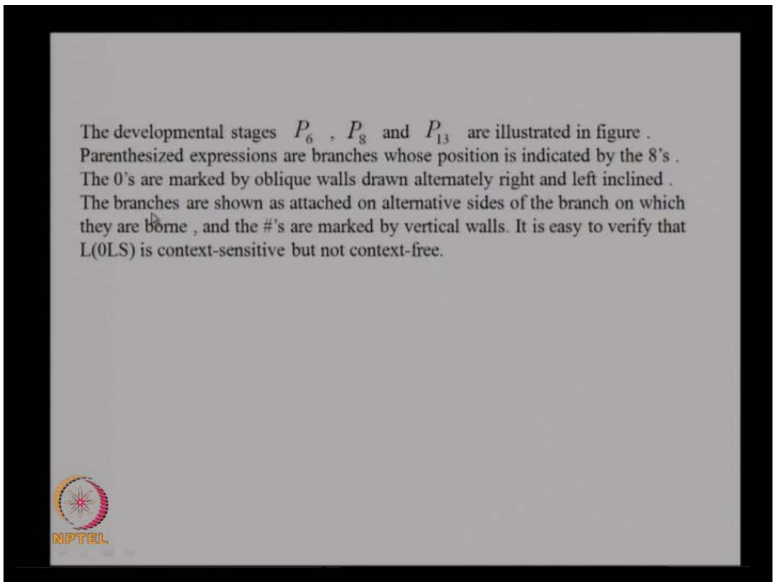
$P_6 = 2\#2\#8(1)7060504$

It can be verified inductively that for all $n \geq 0$,

$P_{n+6} = 2\#2\#8(P_n)08(P_{n-1})0.....08(P_0)07060504.$

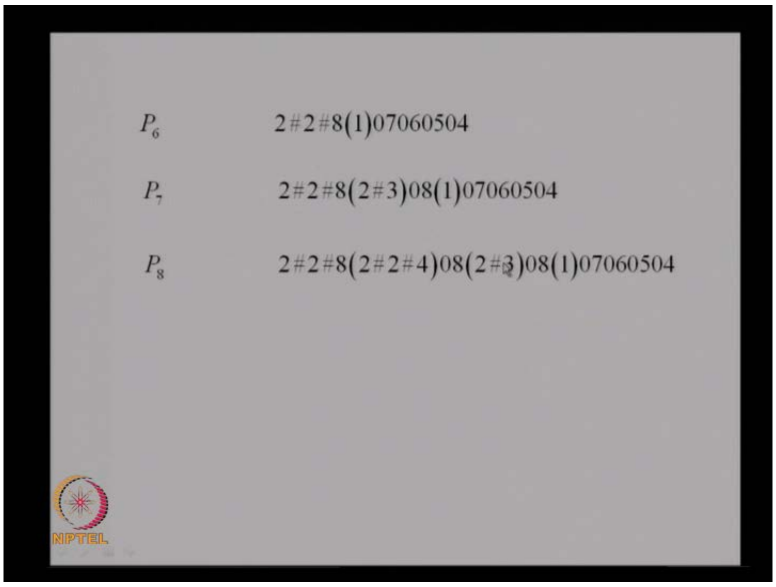
Then you find that at the first step you have 1, then the next step 1 goes to2 hash 3 then the next step you go get 2 hash 2 hash 4 and so on. In the sixth step you get this now, in the n plus sixth step for al n greater than or equal to 0. The n plus sixth step, the string will be of this form P n P n minus 1 etcetera and in between you have these symbols.

(Refer Slide Time: 11:26)


The developmental stages P_6 , P_8 and P_{13} are illustrated in figure . Parenthesized expressions are branches whose position is indicated by the 8's . The 0's are marked by oblique walls drawn alternately right and left inclined . The branches are shown as attached on alternative sides of the branch on which they are borne , and the #'s are marked by vertical walls. It is easy to verify that L(0LS) is context-sensitive but not context-free.

So, the development stages can be represented again as, the growth of a plant and branching out P 6, P 8, P 13 are given in the figure see the figure P 6. We will see it later, the parenthesis expressions are branches whose position is indicated by the 8. When you have a eight, a branch will start from that position. And the branches are shown as attached on alternative sides of the main branch. And hashs are marked by vertical walls, then zeros are marked with the slanting position alternatively mark this way and that way.

(Refer Slide Time: 12:16)



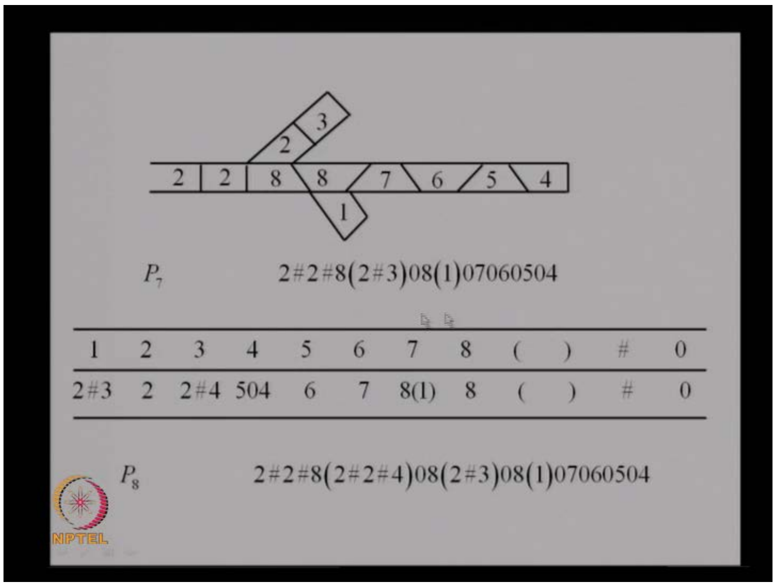
P_6 2#2#8(1)07060504

P_7 2#2#8(2#3)08(1)07060504

P_8 2#2#8(2#2#4)08(2#3)08(1)07060504

So, look at the string for p 6 this is p 6, then the string for p 7, at the 7 instance you get this string, at the 8 instance you get this string.

(Refer Slide Time: 12:30)



P_7 2#2#8(2#3)08(1)07060504

1	2	3	4	5	6	7	8	(	)	#	0
2#3	2	2#4	504	6	7	8(1)	8	(	)	#	0

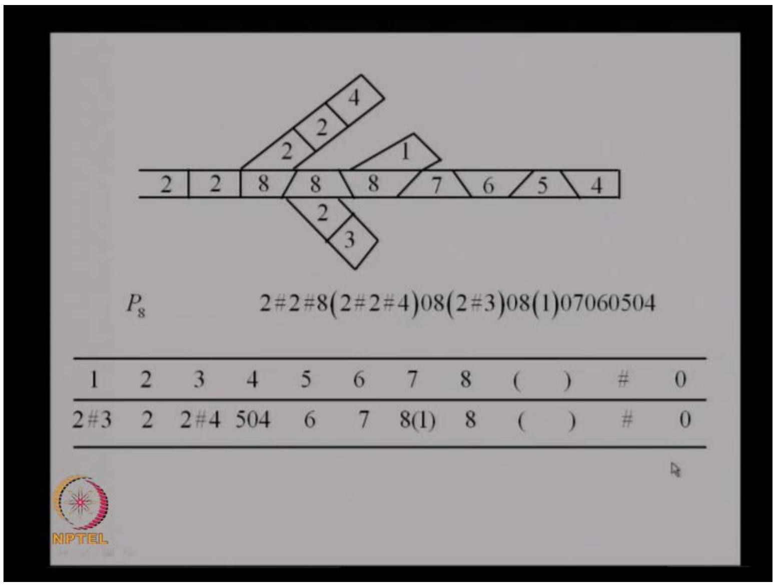
P_8 2#2#8(2#2#4)08(2#3)08(1)07060504

So, look at p 6 is the 2 hash 2 hash etcetera. So, the hash is represented a by a vertical wall in the stem or a plant you can stem like plant. And 8 is where the branches, branch out you have a branch which is mentioned within the parenthesis. So, one is this there is a branch is growing from this plantm then the zeros are represented by oblique walls. So, you have between 0 7 6 5 4, the labels of these cells are given as 7 6 5 4.

Now, this is P 6 now from P 6 how do you get P 7 these are the rules. So, 2 goes to 2 hash goes to hash, 2 goes to 2 hash goes to hash 8 goes to 8 let parenthesis goes to left parenthesis you see this. 1 goes to 2 hash 3 so at this stage from one you get 2 hash 3 right parenthesis goes to right parenthesis 0 goes to 0. But 7 goes to 8, 1 so at this stage you have from 7 8 1 6 goes to 7 5 goes to 6 and 4 goes to 5 0 4. So, the string obtained in the next step is this.

And you see it is represented by a little bit grown steam like this stem there are two eights and in when you have a 8 a branch starts. This branch which as say earlier just 1 has grown like this it has grown into 2 and 3. And another branch has started growing here that is this now this is P 7 from this again the applying the rules 2 goes to 2 hash goes to hash etcetera you get this string, 2 hash 3 will go to 2 goes to 2 hash goes to hash. But 3 will go to 2 hash 4 so we have 2 hash 4 and 1 goes to 2 hash 3 again 7 goes to 8 within parenthesis 1.

(Refer Slide Time: 15:07)

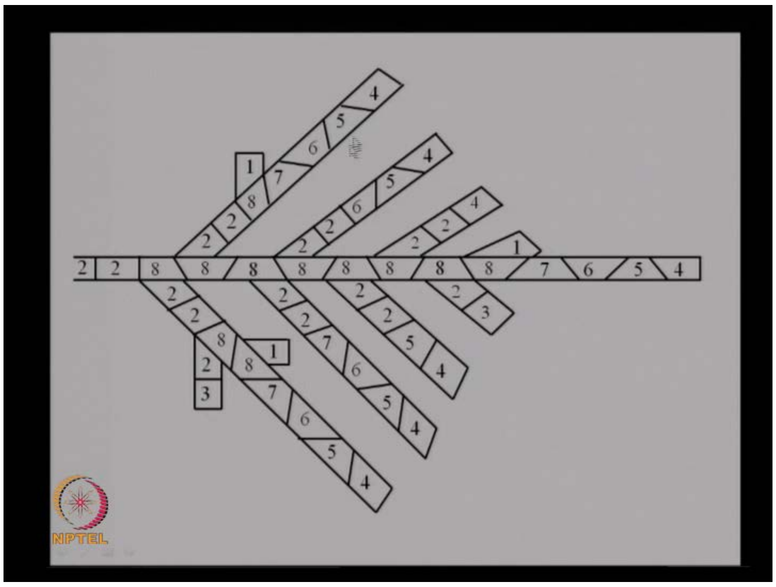


P_8 2#2#8(2#2#4)08(2#3)08(1)07060504

1	2	3	4	5	6	7	8	(	)	#	0
2#3	2	2#4	504	6	7	8(1)	8	(	)	#	0

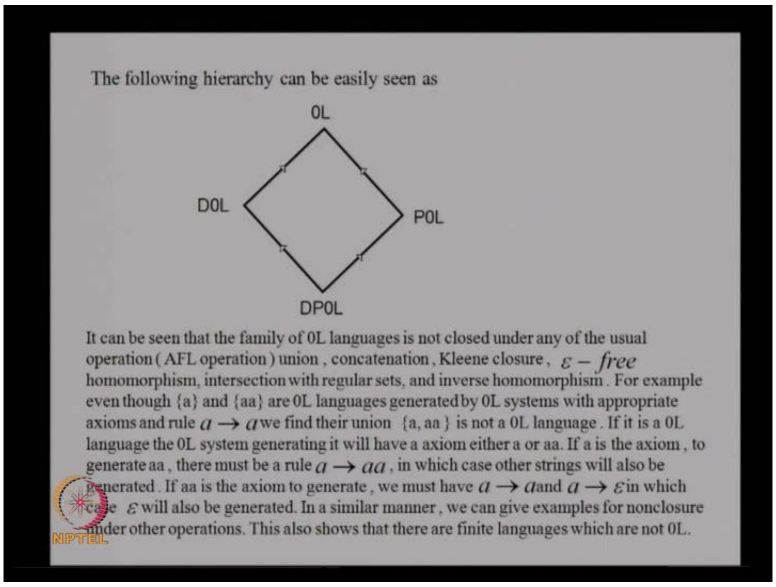
So, at the next instance you get this string so when you get this string see that at the places where you have 8 a branch is started. So, the earlier branch has grown into this now the earlier branch in the previous step has grown like this. A new branch has come at this position. So, the earlier branches have grown a new branch has come and this process is repeated the branches themselves will start creating sub branches.

(Refer Slide Time: 15:45)



And at the thirteenth step you have a very long string and that represent a structure like this you have the main stem from which you have the branches. The first branch has grown into this and it has developed two sub branches. The second branch has developed one sub branch and other branches have been generated and they have grown like this. So, a growth of a plant can be represented in this manner and this was the main intension of lindenmayer in generating this system or in defining this system.

(Refer Slide Time: 16:24)



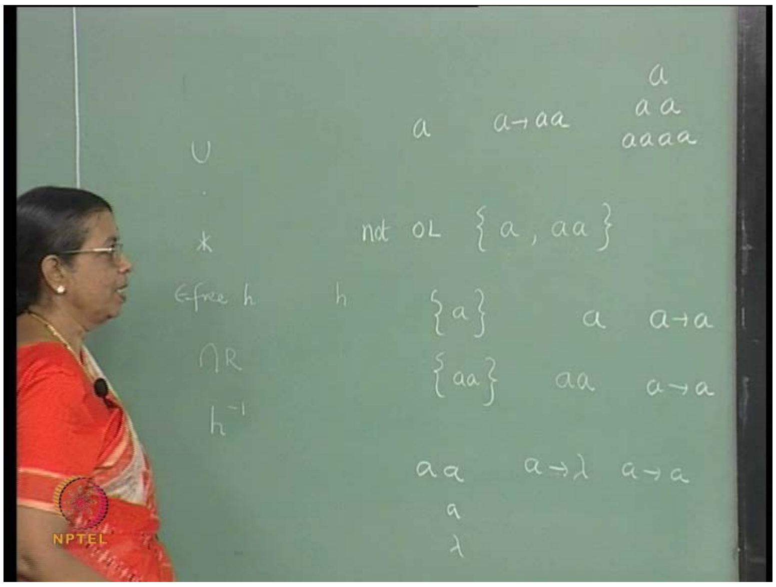
The following hierarchy can be easily seen as

$$\begin{tikzcd}
 & 0L & \\
D0L \arrow[ur] & & P0L \arrow[ul] \\
 & DP0L \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

It can be seen that the family of 0L languages is not closed under any of the usual operation (AFL operation) union , concatenation , Kleene closure , $\varepsilon - free$ homomorphism, intersection with regular sets, and inverse homomorphism . For example even though {a} and {aa} are 0L languages generated by 0L systems with appropriate axioms and rule $a \rightarrow a$ we find their union {a, aa } is not a 0L language . If it is a 0L language the 0L system generating it will have a axiom either a or aa. If a is the axiom , to generate aa , there must be a rule $a \rightarrow aa$, in which case other strings will also be generated . If aa is the axiom to generate , we must have $a \rightarrow a$ and $a \rightarrow \varepsilon$ in which case ε will also be generated. In a similar manner , we can give examples for nonclosure under other operations. This also shows that there are finite languages which are not 0L.

Now, the hierarchy within this is given by this, all DP0L languages are D0L languages as well. And P0L languages as well and they are included in the family of 0 L languages. Now, from formal language theory point of view the main interest in those days was to consider whether a new system is closed under operation and especially six operations have been considered.

(Refer Slide Time: 16:59)

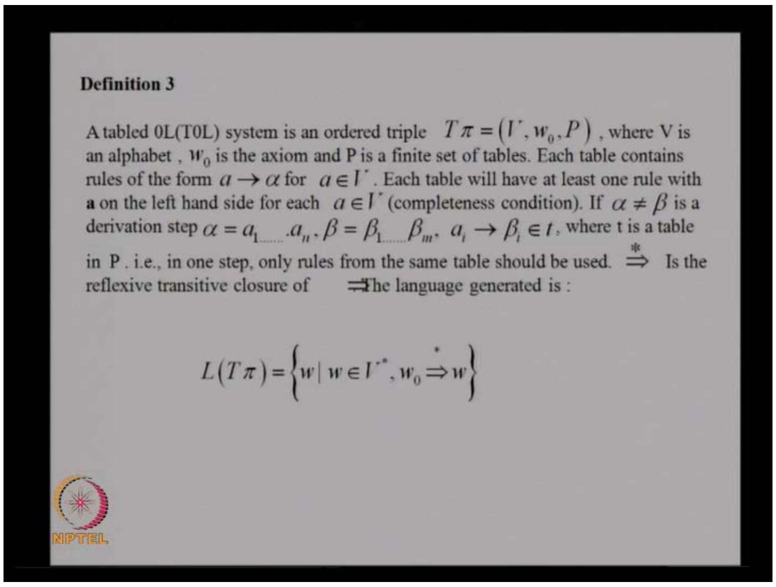


They are union concatenation clinic closure epsilon free h and sometimes ordinary h interception with regular set inverse homomorphism. If a family is closed under all these operations it is called a abstract family of languages. Now, we find that this 0 L languages are not closed under any of these operations look at this language a a which has got just two strings a a.(No audio frtom 17:41 to 17:48) Now, a alone is a 0 L language you can just have an axiom a and a rule a goes to a this will generate only a. a a alone is a 0 L language with axiom a a and the rule a goes to a a this will generate only this string.

But when you consider the union the union has two strings, a and a a it cannot be generated by any 0 L system not 0 L this is not a 0 L language. By we can have only one axiom so if i take a as the axiom I should have a rule of the form a goes to a a to generate this. And if I have this rule then from a I will get a a I can still use the rule and get a a a a and so on the language generated would be a power two power n and not this. So, this is not possible suppose I start with the axiom a a to get a I have to use the rule a goes to lambda also a goes to a. So, from a a I can apply this rule for one a this rule for another a and get a, but if I do that again I can use this rule and get lambda also in the language, but lambda is not here so this way also it is not possible.

And so this is not a 0 L language similarly, you can prove that the family is not closed under any of the other operations as well. So, it is called an anti A F L this was the interest from the formal language theory of you,

(Refer Slide Time: 19:58)



Definition 3

A tabled 0L(T0L) system is an ordered triple $T\pi = (V, w_0, P)$, where V is an alphabet, w_0 is the axiom and P is a finite set of tables. Each table contains rules of the form $a \to \alpha$ for $a \in V$. Each table will have at least one rule with **a** on the left hand side for each $a \in V$ (completeness condition). If $\alpha \neq \beta$ is a derivation step $\alpha = a_1 \ldots a_n, \beta = \beta_1 \ldots \beta_m, a_i \to \beta_i \in t$, where t is a table in P. i.e., in one step, only rules from the same table should be used. $\overset{*}{\Rightarrow}$ Is the reflexive transitive closure of $\Rightarrow$The language generated is :

$$L(T\pi) = \left\{ w \mid w \in V^*, w_0 \overset{*}{\Rightarrow} w \right\}$$

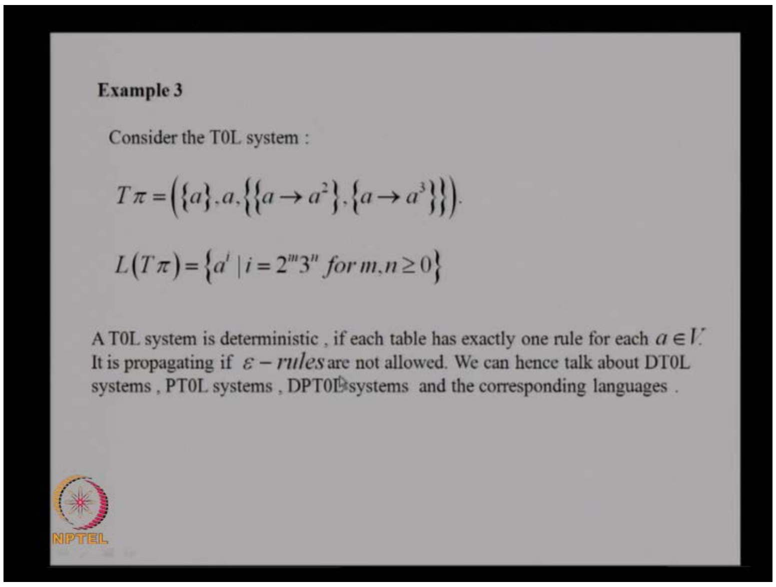
in order to overcome that extended systems were defined we will come to that in a moment. The table to 0 L system was defined what is a table to 0 L system actually the motivation for this was to describe the growth of plants mainly plants any organism for that matter. And when you want to describe the growth of a plant they may follow a certain set of rules during daytime. Because in the presence of sunlight something hapsence and in the absence of sunlight something else happens.

So, the growth mechanism may be different during daytime and it may be different during night. So, you may require two sets of rules to describe the growth of a plant alos the growth of a plant will differ during summer during winter actually they may shed all the leaves during autumn. And the in spring the leaves start coming and in winter they only have stems such a thing happens so during different seasons different things happen in a plant. And the rules of growth will be different during each season so you may have to have different sets of rules to describe the growth of such a plant. And that is why we have what is known as a table 0 L system. Lindenmayer defined this to describe the growth in different season and different time during a day.

A table 0 L systems is an ordered triple like this where it is denoted as a t pi where v is the alphabet naught is the axiom P is the set of tables. Now, each table contains rules of the form a goes to alpha where on the left hand side you have a single symbol and on the right hand side you have a string. Each table will have at least one rule with a on the left hand side for each a belonging to v this is called the completeness condition. And if you have only one rule it is called a deterministic system. Now, suppose alpha is a one a two a n then from alpha you can derive beta by this a one will be replaced by beta 1 a two will b replaced by beta two and so on.

But all these rules a I goes to beta I should be from the same table that is at a particular step. You can use only rules from the particular table. Where t is a table only rule of the same table should be used at a particular step. Now, as usual double arrow star will represent the reflexive transitive closure of this the language generated is denoted as w w belongs to v star w naught derives w. That is starting from the axiom you should be able to derive a set of strings if w is derivable from the axiom then w belongs to the language generated by the system.

(Refer Slide Time: 23:55)



Example 3

Consider the T0L system :

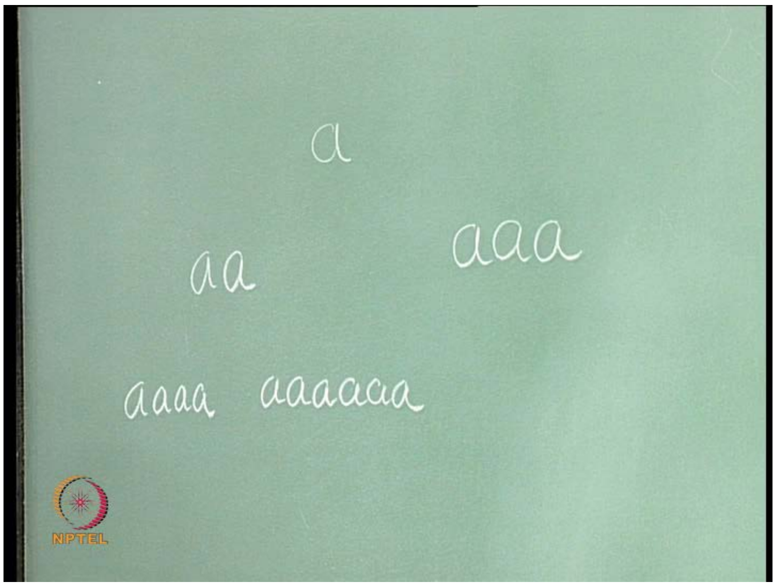
$T\pi = \left(\{a\}, a, \{\{a \to a^2\}, \{a \to a^3\}\}\right)$.

$L(T\pi) = \{a^i \mid i = 2^m 3^n \ for\ m,n \geq 0\}$

A T0L system is deterministic , if each table has exactly one rule for each $a \in V$. It is propagating if $\varepsilon - rules$ are not allowed. We can hence talk about DT0L systems , PT0L systems , DPT0L systems and the corresponding languages .

Consider this example you have only one symbol a which is the axiom and two tales are there. In one table you use the rule a goes to a squared in another table you use the rule a goes to a cubed. So, suppose I start with a the next step if I use this table I will get a squared if I use this table I will get a cubed. So, a squared also belongs to the language a cubed also belongs to the language. Now if I have a a then there are possibilities like this.

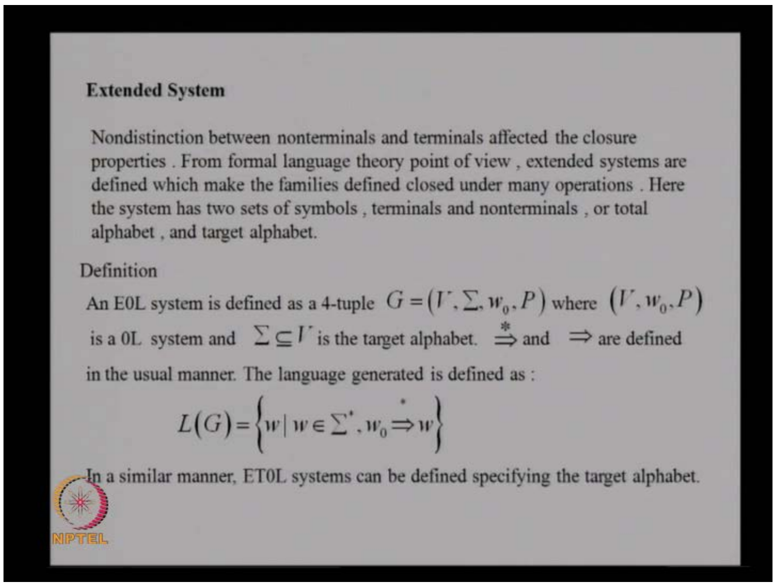
(Refer Slide Time: 24:41)



Starting with a if I use one table I get a a if I use another table I may get a a now for this again if I use rule from the table a goes to a a I may get a a a a. I should use the same table for this a and this a if I use the rule a goes to a cubed for this a a. I shall get a a a a a a, but I cannot apply a rule from one table for this a and a rule for another table for this a. I have to use the rule from the same table so if you look carefully you will find that the number of as will be doubled in the next step or tripled in the next step.

So, in general you will get a string of the form a power i where i will b of the form 2 power m 3 power n. Again we can define deterministic systems propagating systems, and so on now we have tables if each table exactly one rule for ev ery a belonging to the alphabet. Then you call it as a deterministic system if you do not have epsilon rules then it is a propagating system. So, you have deterministic table to 0 L system propagation table to 0 L system deterministic and propagating table to 0 L systems.

(Refer Slide Time: 26:28)



Extended System

Nondistinction between nonterminals and terminals affected the closure properties . From formal language theory point of view , extended systems are defined which make the families defined closed under many operations . Here the system has two sets of symbols , terminals and nonterminals , or total alphabet , and target alphabet.

Definition

An E0L system is defined as a 4-tuple $G=(V,\Sigma,w_0,P)$ where (V,w_0,P) is a 0L system and $\Sigma\subseteq V$ is the target alphabet. $\overset{*}{\Rightarrow}$ and $\Rightarrow$ are defined in the usual manner. The language generated is defined as :

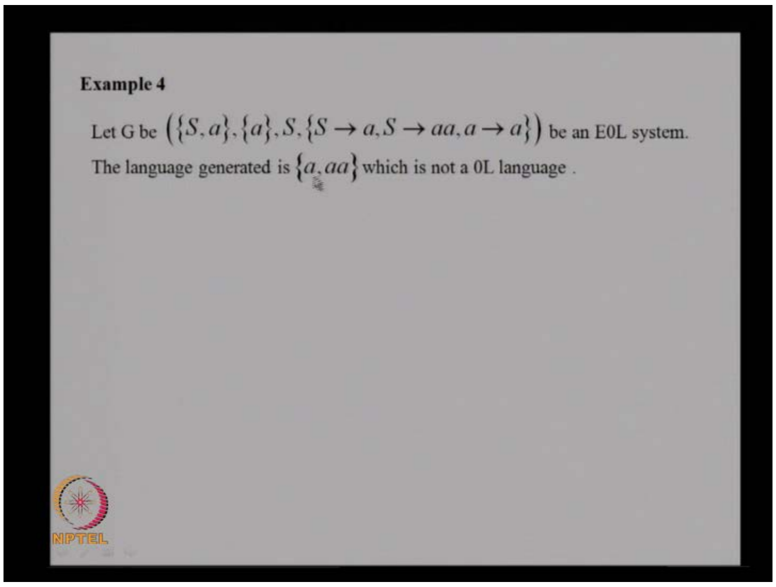
$$L(G)=\left\{w \mid w\in\Sigma^*, w_0\overset{*}{\Rightarrow}w\right\}$$

In a similar manner, ET0L systems can be defined specifying the target alphabet.

The main defect from formal language theory point of view is the non closure under operations. Table 0 L systems were also closed under many operations so in order to make the system closed under many of the operations. The distinction between non terminals and terminals was introduced actually this is the one which creates the non closure property. So, non distinction between non terminals and terminals are affected the closure properties so from formal theory point of view extended systems have been defined. Which make the families closed under many operations, here we have two sets of symbols terminals and non terminals or instead you consider. The total alphabet and the subset of it as a target alphabet target alphabet corresponds to the terminals.

And the total alphabet corresponds to both terminals and non terminals or the union of terminals and non terminals a e 0 L system or a extended 0 L-system is defined as a four tuple. v the total alphabet sigma the target alphabet which is the subset of v. And an axiom w naught a set of production p similar derivation steps are defined at every step you have to parallelly rewrite the symbols.Then double arrow star is the reflexive transitive closure and the language consist of strings, which belong to the target alphabet. That is from the axiom it should be possible to derive the string w where w belongs to sigma star. If it contains a symbol which is not in the target alphabet that string will not belong to the language. Again within this also we can define tables and so you have E T 0 L systems.

(Refer Slide Time: 28:38)



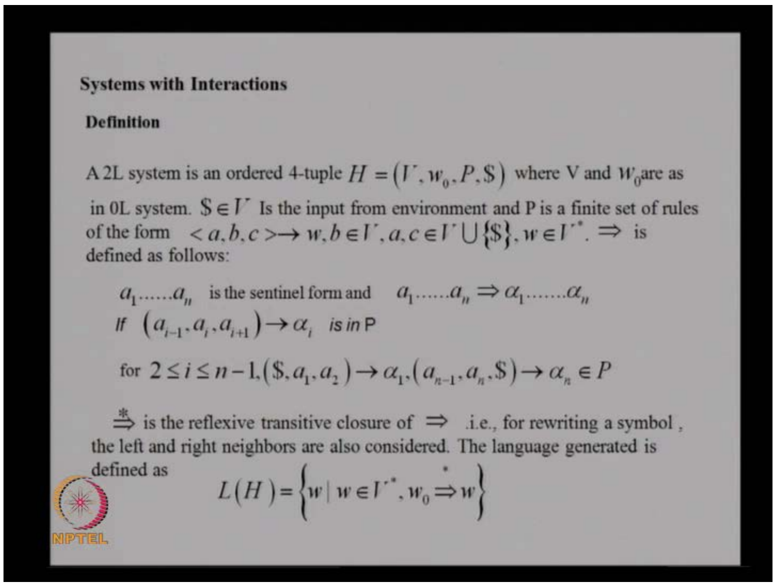
Example 4

Let G be $(\{S,a\},\{a\},S,\{S\rightarrow a,S\rightarrow aa,a\rightarrow a\})$ be an E0L system.

The language generated is $\{a,aa\}$ which is not a 0L language .

For example, tale this start from s s is the axiom you have rules s goes to a s goes to a a, a goes to a. So, you start with this you will get a you start with this you will get a a then you can apply this rule and no more strings will be generated. Now, S will not be in the language the axiom will not be in the language because it is not in the target alphabet. But a will be in the language a a will be in the language and so this is a extended 0 L language you see that we saw that this is not a 0 L language. So, but it is a E 0 L language.

(Refer Slide Time: 29:24)



Systems with Interactions

Definition

A 2L system is an ordered 4-tuple $H = (V, w_0, P, \$)$ where V and w_0 are as in 0L system. $\$ \in V$ Is the input from environment and P is a finite set of rules of the form $< a, b, c > \rightarrow w, b \in V, a, c \in V \cup \{\$\}, w \in V^*$. $\Rightarrow$ is defined as follows:

$a_1 \ldots\ldots a_n$ is the sentinel form and $a_1 \ldots\ldots a_n \Rightarrow \alpha_1 \ldots\ldots \alpha_n$

If $(a_{i-1}, a_i, a_{i+1}) \rightarrow \alpha_i$ is in P

for $2 \le i \le n-1, (\$, a_1, a_2) \rightarrow \alpha_1, (a_{n-1}, a_n, \$) \rightarrow \alpha_n \in P$

$\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$ .i.e., for rewriting a symbol , the left and right neighbors are also considered. The language generated is defined as

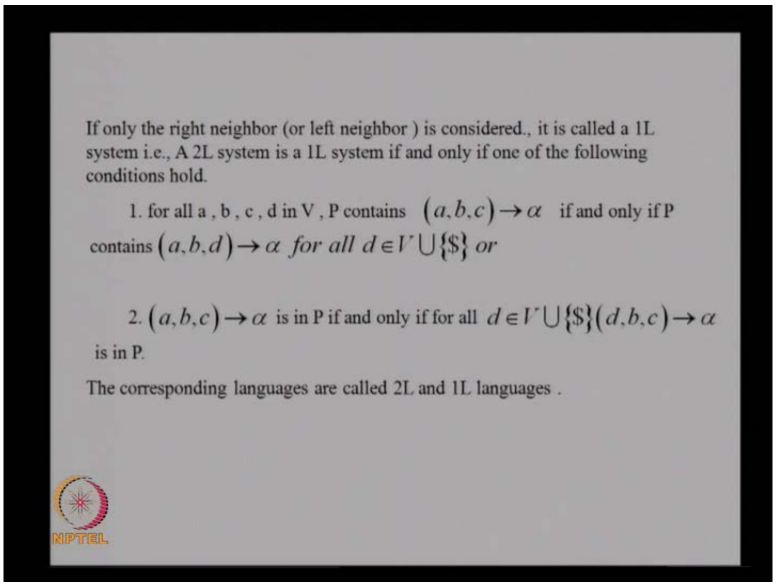
$$L(H) = \left\{ w \mid w \in V^*, w_0 \overset{*}{\Rightarrow} w \right\}$$

Now, so far we have considered systems without interaction you can also talk about systems with interaction. What is interaction on the left hand side you have a and a b in a string a b c occurs like this. To the towards a left of a b a occurs and to the right of b c occurs then v is replaced by w. That is replacing b by w depends not only on b, but the left neighbor a and the right neighbor c. Something known as cellular automata is very is similar to that the state changes in a cellular automata occur like this.

I will not go into depth in the system so the 2 0 L system everything depends on the left neighbor and the right neighbor. So, when you have a string like this a 1 a 2 a n on the left of a you have the environment. The environment is denoted by a dollar symbol that is why you have one more component in the system. So, the first symbol the rule will depend on the environment and the right neighbor. The second symbol the rule will depend on the left and the right neighbor and so on for the last symbol a n.

The rule will depend on the left neighbor and the environment the rule will be of this form. So, a one will be replaced by alpha one a two will be replaced by alpha two and so on so you get from a one a two a n alpha one alpha two alpha n. And the rewriting is completely parallel the language generated again is the set of strings, which are derivable from the axiom. Sometimes the growth of the plant cannot be explained in a proper manner just by rules without interaction. But if you have interaction or rules with interaction then the growth can be explained in a much easier manner.

(Refer Slide Time: 30:52)



If only the right neighbor (or left neighbor) is considered., it is called a 1L system i.e., A 2L system is a 1L system if and only if one of the following conditions hold.

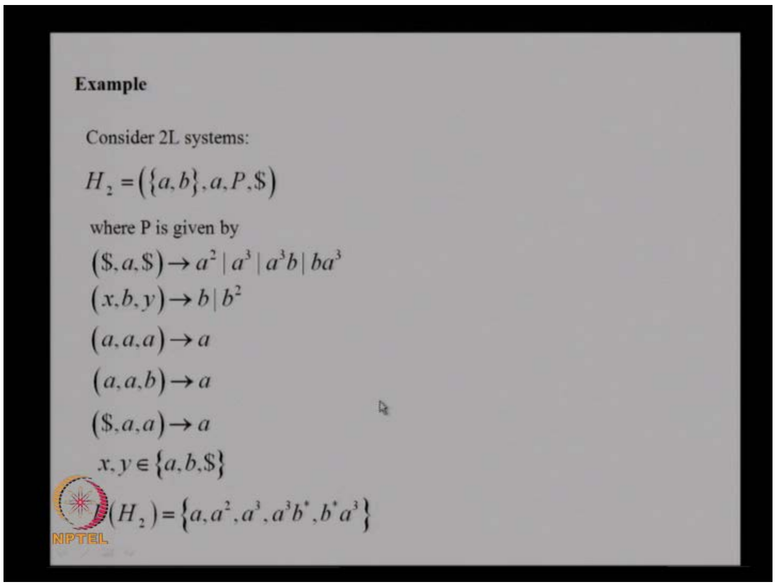
1. for all a , b , c , d in V , P contains $(a,b,c)\rightarrow\alpha$ if and only if P contains $(a,b,d)\rightarrow\alpha$ *for all* $d\in V\cup\{\$\}$ *or*

2. $(a,b,c)\rightarrow\alpha$ is in P if and only if for all $d\in V\cup\{\$\}(d,b,c)\rightarrow\alpha$ is in P.

The corresponding languages are called 2L and 1L languages .

Now, if you have just on one side the rule it is called a 1 L system. See either you have only considered the left neighbor or you only considered the right neighbor it is called a 1 L system.

(Refer Slide Time: 32:05)



Example

Consider 2L systems:

$H_2=(\{a,b\},a,P,\$)$

where P is given by

$(\$,a,\$)\rightarrow a^2\mid a^3\mid a^3b\mid ba^3$

$(x,b,y)\rightarrow b\mid b^2$

$(a,a,a)\rightarrow a$

$(a,a,b)\rightarrow a$

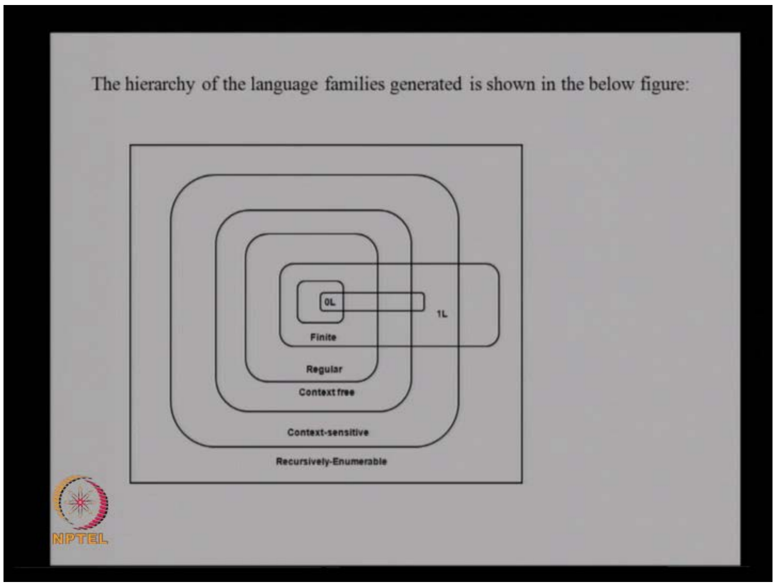
$(\$,a,a)\rightarrow a$

$x,y\in\{a,b,\$\}$

$(H_2)=\{a,a^2,a^3,a^3b^*,b^*a^3\}$

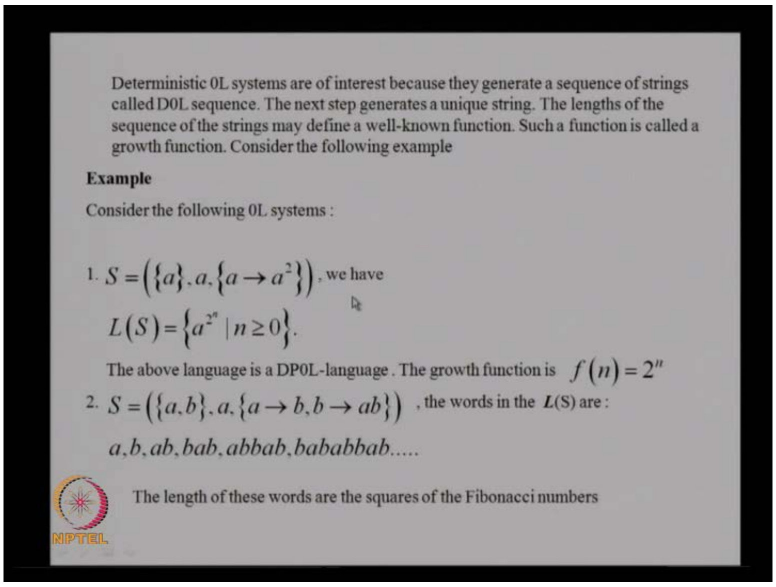
Again some examples, are given here so the system has an axiom a tewo symbols a b rules are given like this. So, starting with a you have dollar and dollar and then you can replace it by a squared a cubed a cubed b etcetera. Then when you have a a b then the rule will be of this form b can be go to b or b squared. So, this sort of a language will be generated by this system.

(Refer Slide Time: 32:39)


The hierarchy of the language families generated is shown in the below figure:

The hierarchy between this and the class were 0 L languages are include is given by this. This is a system you find that 0 l does not include some finite sets also a a is the finite set, but it is not a 0 L language. But some context sensitive languages like a power two power n are there so the class of zero l languages like this. The class of one l languages are like this where this is the chomskyan hierarchy.

(Refer Slide Time: 33:18)


Deterministic 0L systems are of interest because they generate a sequence of strings called D0L sequence. The next step generates a unique string. The lengths of the sequence of the strings may define a well-known function. Such a function is called a growth function. Consider the following example

Example

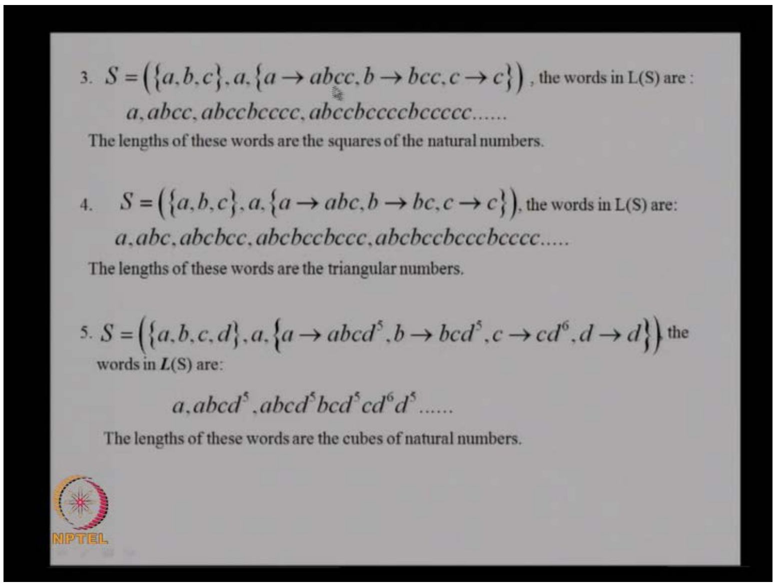
Consider the following 0L systems :

1. $S=(\{a\}, a, \{a\to a^2\})$, we have
   $L(S)=\{a^{2^n} \mid n\ge 0\}$.
   The above language is a DP0L-language . The growth function is $f(n)=2^n$
2. $S=(\{a,b\}, a, \{a\to b, b\to ab\})$, the words in the L(S) are :
   $a, b, ab, bab, abbab, bababbab.....$
   The length of these words are the squares of the Fibonacci numbers

Some of these rules are like this consider this system it is a deterministic system. Now, we will consider only deterministic system where you have only one rule for each symbol. You have a goes to a squared then at the nth step you get a power two power n the length of the string is given by two power n and this is called the growth function in this case. So, consider this rule or this system. Initially you have a as the axiom next step it goes to b.

Next step it is rewritten as a b. Then next step a is rewritten as b b, is rewritten as a b and so on. If you look at the length of the strings this is zeroth step one first step second step and so on. The length of the string here is five it is the sum of the previous two steps the length of the string is eight here which is the sum of the length of these two strings. So, you see that the length of these words are just fibonacci sequence fibonacci numbers.

(Refer Slide Time: 34:36)



3. $S = (\{a,b,c\}, a, \{a \rightarrow abcc, b \rightarrow bcc, c \rightarrow c\})$, the words in L(S) are :

$a, abcc, abccbcccc, abccbccccbcccccc......$

The lengths of these words are the squares of the natural numbers.

4. $S = (\{a,b,c\}, a, \{a \rightarrow abc, b \rightarrow bc, c \rightarrow c\})$, the words in L(S) are:

$a, abc, abcbcc, abcbccbccc, abcbccbcccbcccc.....$

The lengths of these words are the triangular numbers.

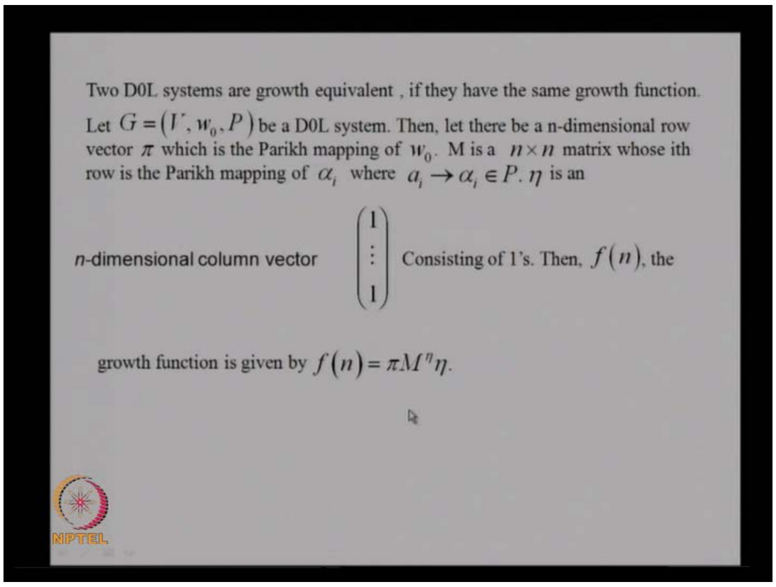
5. $S = (\{a,b,c,d\}, a, \{a \rightarrow abcd^5, b \rightarrow bcd^5, c \rightarrow cd^6, d \rightarrow d\})$ the words in L(S) are:

$a, abcd^5, abcd^5bcd^5cd^6d^5......$

The lengths of these words are the cubes of natural numbers.

And if you look at this one, you find that the sequence generated is a goes to a b c c, then again replacing a by this rule b by this rule c by this rule you find you get this. The lengths are one four nine sixteen and so or they are squares of the natural numbers. Similarly, if you considered this system the sequence generated is this, one three six ten etcetera. The length of the sequences are one one plus two one plus two plus three one plus two plus three plus four and so on they are called triangular numbers. The length of the n string will be sigma i is equal to one to n. Now, another system is given like this if you look into the length of the strings, it is one eight twenty seven and so on the length of the words are cubes of natural numbers.

(Refer Slide Time: 35:45)



Two D0L systems are growth equivalent , if they have the same growth function.

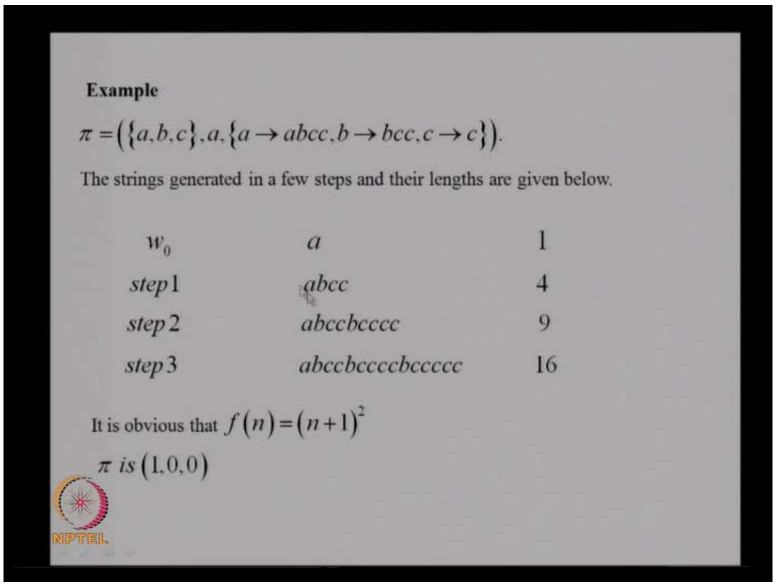
Let $G = (\Gamma, w_0, P)$ be a D0L system. Then, let there be a n-dimensional row vector π which is the Parikh mapping of w_0. M is a $n \times n$ matrix whose ith row is the Parikh mapping of α_i where $a_i \to \alpha_i \in P$. η is an

n-dimensional column vector $\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ Consisting of 1's. Then, $f(n)$, the

growth function is given by $f(n) = \pi M^n \eta$.

The whole thing can be explain in a convenient manner.

(Refer Slide Time: 35:47)



Example

$\pi = (\{a,b,c\}, a, \{a \to abcc, b \to bcc, c \to c\})$.

The strings generated in a few steps and their lengths are given below.

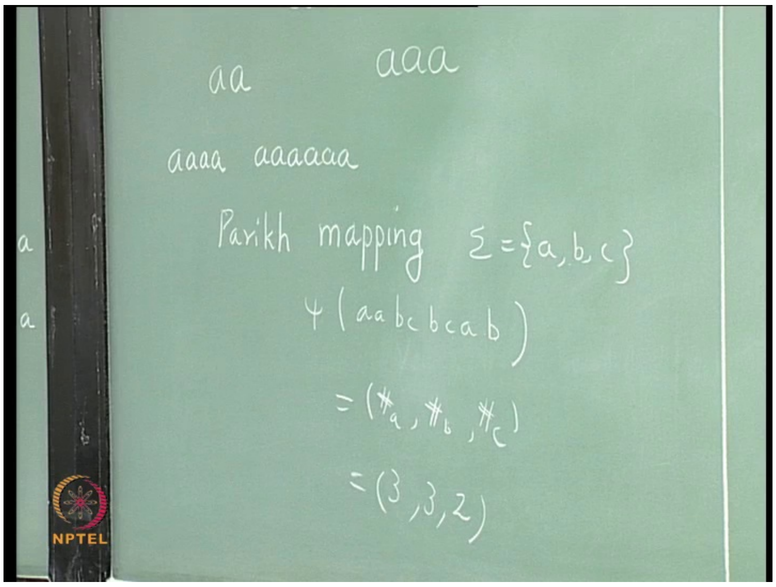
w_0	a	1
$step 1$	$abcc$	4
$step 2$	$abccbcccc$	9
$step 3$	$abccbccccbccccc$	16

It is obvious that $f(n) = (n+1)^2$

$\pi\ is\ (1,0,0)$

Let me take this particular example, the axiom is a and the rules are like this so the lengths of the sequences are given like this.

(Refer Slide Time: 36:08)



aa aaa

aaaa aaaaaa

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

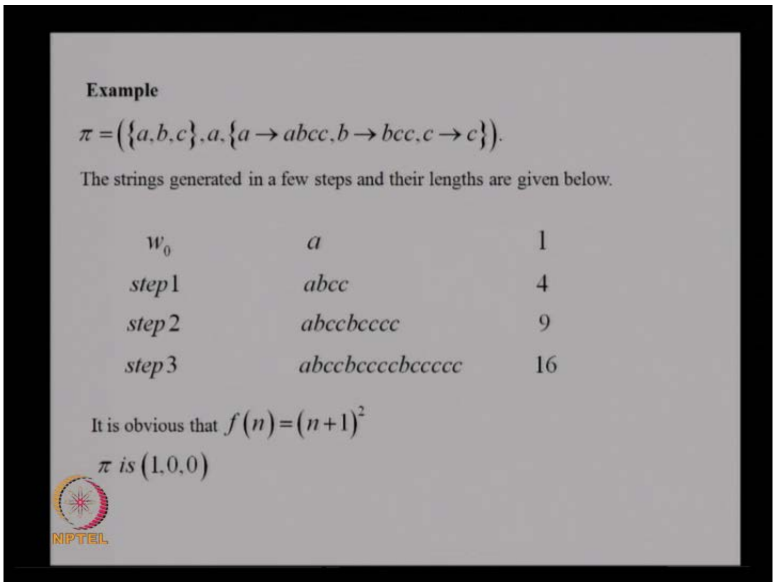
$\psi(aabcbcab)$

$= (\#_a, \#_b, \#_c)$

$= (3, 3, 2)$

Now, the parikh mapping of a string is like this Parikh mapping; suppose I have a string a a b I will take the alphabet here. Some string a a b c b c a b something like this; the parikh mapping of this is denote by this it is a vector where the elements of integers. The first one denote the number of a(s) so I assume the order a b c for symbols in sigma. The second denotes the number of b(s) the third denotes the number of c(s). So, in this example it will be number of as will be one, two, three. The number of b(s) will be one 2 3 the number of c(s) is 2, so the parikh mapping of this string is 3, 3, 2.

(Refer Slide Time: 37:23)



Example

$\pi = (\{a,b,c\}, a, \{a \to abcc, b \to bcc, c \to c\})$.

The strings generated in a few steps and their lengths are given below.

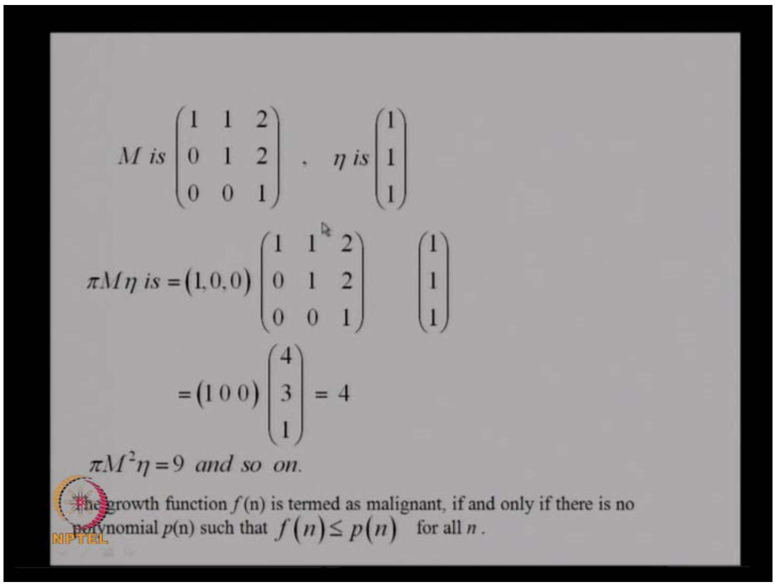
w_0	a	1
$step\,1$	$abcc$	4
$step\,2$	$abccbcccc$	9
$step\,3$	$abccbccccbccccc$	16

It is obvious that $f(n) = (n+1)^2$

π *is* $(1,0,0)$

Now this idea if you use.You find a a connection between matrix algebra.

(Refer Slide Time: 37:25)



$$M \text{ is } \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}, \quad \eta \text{ is } \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\pi M \eta \text{ is } = (1,0,0) \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

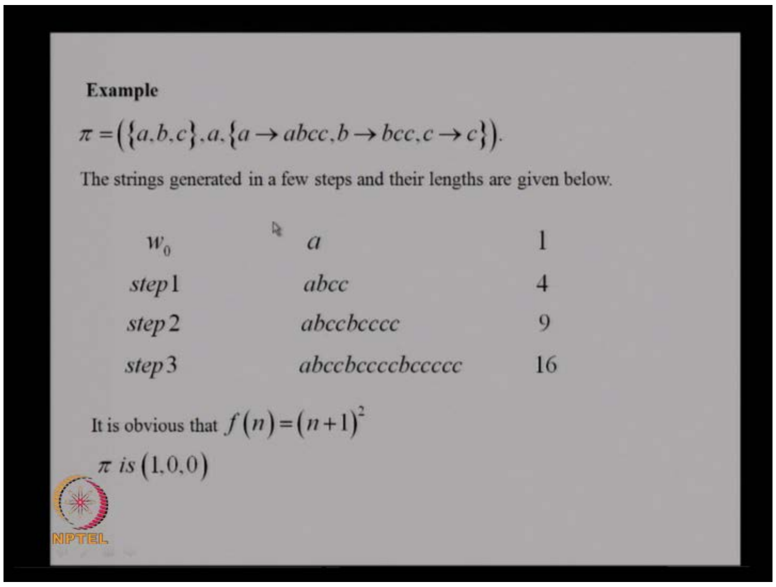
$$= (1\ 0\ 0) \begin{pmatrix} 4 \\ 3 \\ 1 \end{pmatrix} = 4$$

$\pi M^2 \eta = 9$ *and so on.*

The growth function f(n) is termed as malignant, if and only if there is no polynomial p(n) such that $f(n) \leq p(n)$ for all n.

And this M is a n by n matrix, where n is the size of the alphabet.

(Refer Slide Time: 37:32)



Example

$\pi = (\{a,b,c\}, a, \{a \to abcc, b \to bcc, c \to c\}).$

The strings generated in a few steps and their lengths are given below.

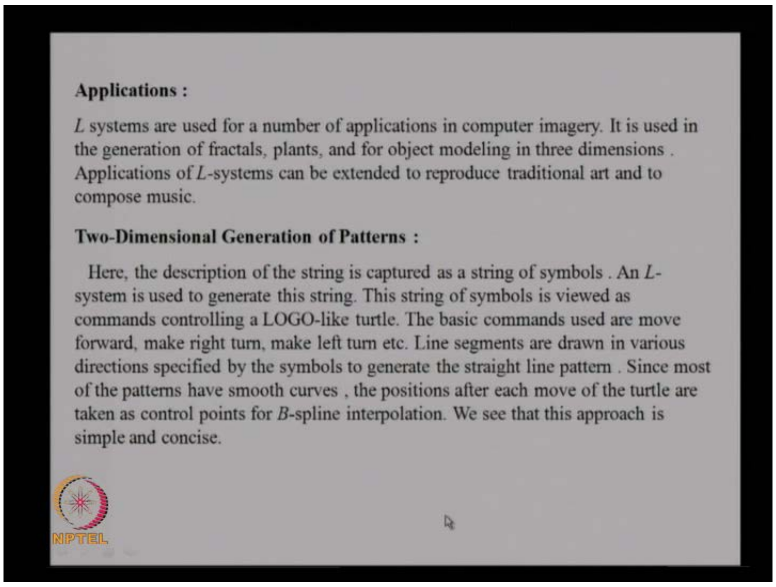
w_0	a	1
$step 1$	$abcc$	4
$step 2$	$abccbcccc$	9
$step 3$	$abccbccccbccccc$	16

It is obvious that $f(n) = (n+1)^2$

π *is* $(1,0,0)$

So, three a b c is three so it is a three by three matrix. The order is a is first then b is second then c is third so if you look at the right hand side further rule a b c c. The rule for a is this and the number of as on the right hand side is one one number of bs is one number of cs is two. The parikh mapping will be one one two and that is the first row. The second symbol is b the parikh mapping of that is 0 1 2, and that is a second row. For c the right hand side is c the parikh mapping of that is 0 0 1, and you have the third row. (No audio from 38:35 to 38:42) The parikh mapping of the axiom is given by this M is known as the growth matrix and if you have a column vector e eta. Which consist of just once pi M eta is pi multiplied by M then by eta it gives you four. That is what you get in the first step so this is a string and this is the length of the string. In the second step in the length of the string is nine if you calculate pi n squared eta that will be nine. So, in general at the nth step the length of the string is given by pi M power n eta. And you can specify the growth function in this manner making use of the parikh vector for the axiom and the growth matrix. Now, the growth function if it is polynomial bounded it is not malignant if it is not bounded by a polynomial it is called malignant.

(Refer Slide Time: 39:48)



Applications :

L systems are used for a number of applications in computer imagery. It is used in the generation of fractals, plants, and for object modeling in three dimensions . Applications of *L*-systems can be extended to reproduce traditional art and to compose music.

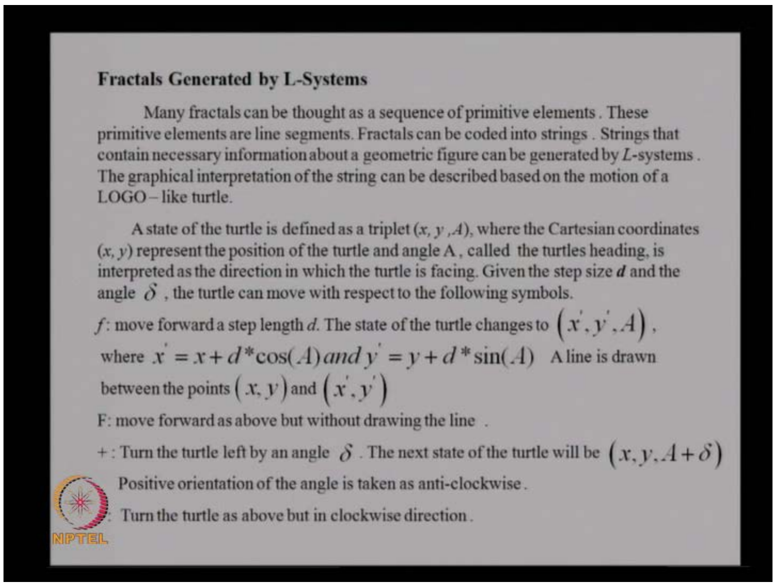
Two-Dimensional Generation of Patterns :

Here, the description of the string is captured as a string of symbols . An *L*-system is used to generate this string. This string of symbols is viewed as commands controlling a LOGO-like turtle. The basic commands used are move forward, make right turn, make left turn etc. Line segments are drawn in various directions specified by the symbols to generate the straight line pattern . Since most of the patterns have smooth curves , the positions after each move of the turtle are taken as control points for *B*-spline interpolation. We see that this approach is simple and concise.

Now, the use of L-systems is really in computer imagery, not only describing the growth of the plants you can describe, picture. And also generate beautiful pictures making use of L-system you have to only write a few rules and the specify the number of iterations and you can get beautiful patterns. Three dimensional objects also can be modeled using this. Now, the description of the string is captured as a string of symbols. And L-system is used to generate the string and then this string of symbols is viewed as commands controlling a logo like turtle. You can have dimensional or three dimensional turtle the basic commands used are move forward by drawing a line.

Make a right turn make a left turn etcetera. Line segments are drawn in various directions specified by the symbols to generate the straight line pattern. Since most of the have smooth curves the positions after each move of the turtle are taken and the controlled points. They are taken as the controlled points for B-spline or b-spline interpolation. That is the position after each move of the turtle are taken as controlled points. And B-spline interpolation is used to draw a smooth curve, this approach is very concise we will see with some examples.

(Refer Slide Time: 41:35)



Fractals Generated by L-Systems

Many fractals can be thought as a sequence of primitive elements. These primitive elements are line segments. Fractals can be coded into strings. Strings that contain necessary information about a geometric figure can be generated by *L*-systems. The graphical interpretation of the string can be described based on the motion of a LOGO – like turtle.

A state of the turtle is defined as a triplet (x, y, A), where the Cartesian coordinates (x, y) represent the position of the turtle and angle A, called the turtles heading, is interpreted as the direction in which the turtle is facing. Given the step size ***d*** and the angle δ, the turtle can move with respect to the following symbols.

f: move forward a step length *d*. The state of the turtle changes to (x', y', A), where $x' = x + d * \cos(A)$ *and* $y' = y + d * \sin(A)$ A line is drawn between the points (x, y) and (x', y')

F: move forward as above but without drawing the line .

+ : Turn the turtle left by an angle δ. The next state of the turtle will be $(x, y, A + \delta)$

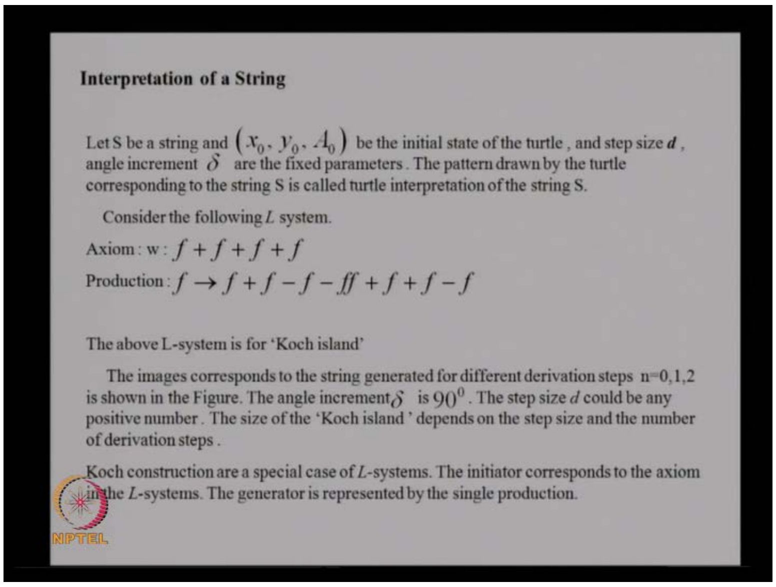
Positive orientation of the angle is taken as anti-clockwise.

: Turn the turtle as above but in clockwise direction.

You can generate beautiful fractals using this. In general the position of the curser turtles is denoted by three point x y or the x y coordinates and a is the orientation of the curser. Then if you move through a distance delta, the step size is d and the angle increment is delta. So, from one point if you move through a distance d then the new position will be x dash y dash a. Where x dash is given by x plus d cos A and y dahs is given by d sin a y plus d sin A. A line is drawn between two points x y and x dash y dash if suppose you denote f. Then it may represent the command to the curser move forward as above without drawing a line. Small f may denote move forward a step length of d by drawing a line, plus may denote turn the turtle left by angle delta.

So, the new position will be x y and then angle will be incremented by delta that is anti clockwise rotation, minus may denote the corresponding clockwise rotation.

(Refer Slide Time: 43:11)



Interpretation of a String

Let S be a string and (x_0, y_0, A_0) be the initial state of the turtle, and step size d, angle increment δ are the fixed parameters. The pattern drawn by the turtle corresponding to the string S is called turtle interpretation of the string S.

Consider the following L system.

Axiom : w : $f + f + f + f$

Production : $f \rightarrow f + f - f - ff + f + f - f$

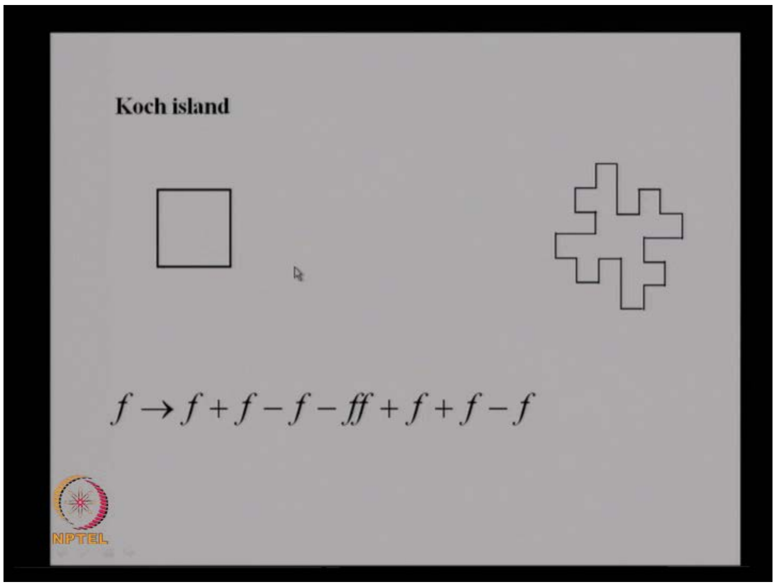
The above L-system is for 'Koch island'

The images corresponds to the string generated for different derivation steps n=0,1,2 is shown in the Figure. The angle increment δ is 90^0. The step size d could be any positive number. The size of the 'Koch island' depends on the step size and the number of derivation steps.

Koch construction are a special case of L-systems. The initiator corresponds to the axiom in the L-systems. The generator is represented by the single production.

So, with this you find that you can interpret a string and then draw a picture. For example, start with this rule with the axiom with this axiom and rule like this. For f we have a rule f goes to f plus f minus f minus f f plus f plus f minus f of course. I have not specified the other rules plus goes to plus minus goes to minus. So, if you look at the axiom.

(Refer Slide Time: 43:48)

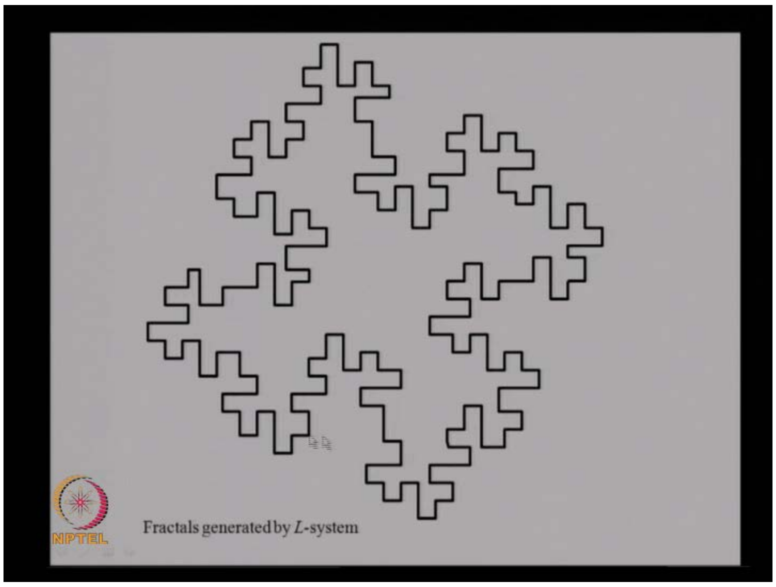


Koch island

$f \rightarrow f + f - f - ff + f + f - f$

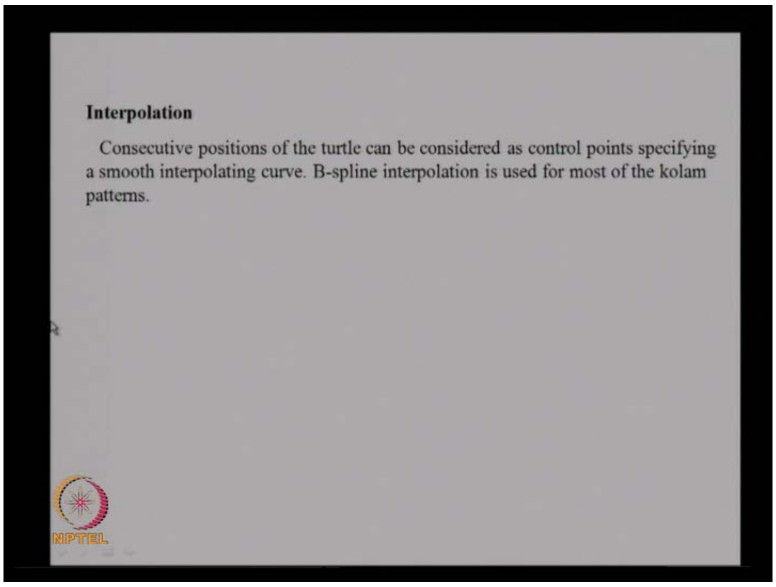
Look at it carefully again, plus f plus f plus f plus f and if f denotes draw a line of unit length. And plus denotes turn anti clockwise through ninety degrees, what does the axiom denote it denotes f plus plus means turn f plus f plus f so you draw a square. Now, you use a rule like this so in the axiom each f will be replaced by this string then let us see what we get. The first f when you replace it start from here, f plus f minus f minus f f f plus f plus f minus f. From this to this replacing one f you will get then again turning anticlockwise through ninety degrees using a plus from this to this position. You will have another f replaced by the string. Then again from this to this position and the last f will be replaced by this. So, you find that at the next stage you get this one.

(Refer Slide Time: 45:32)



And in the next stage you will get this one, if you want to draw a smooth curve it will become the cost island.

(Refer Slide Time: 45:40)

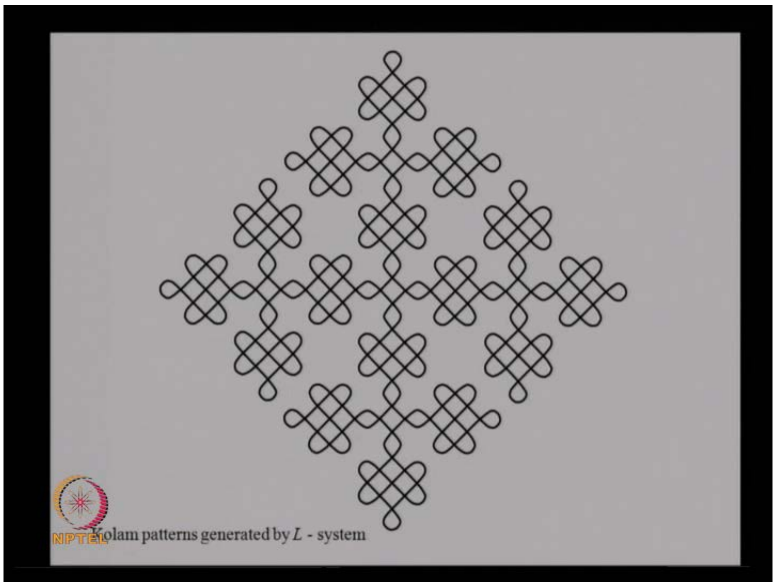


Interpolation

Consecutive positions of the turtle can be considered as control points specifying a smooth interpolating curve. B-spline interpolation is used for most of the kolam patterns.

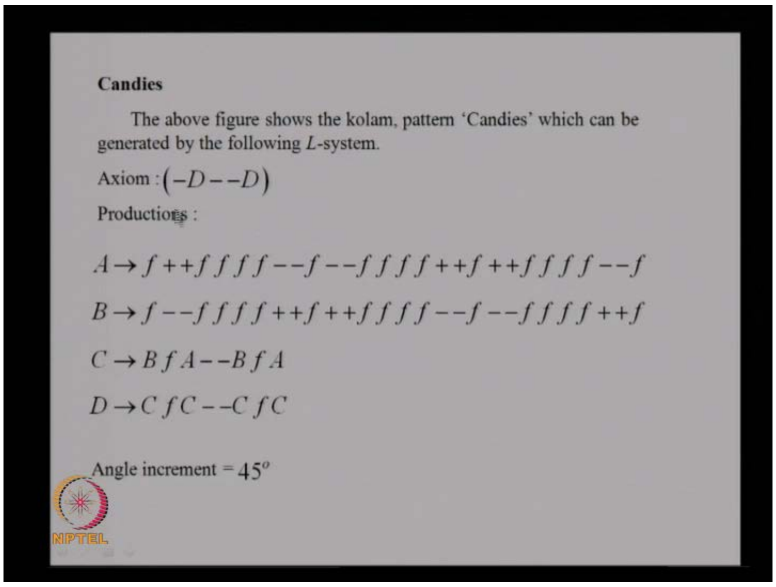
Consecutive positions of the turtle can be considered as control points specifying a smooth interpolating curve. B-spline interpolation is used or most of the kolam patters. Kolam patterns are those patterns which are drawn in front of the house especially in south india. Every morning in villages now a days towns or cities have only one flats and they do not have enough space to draw the patterns. But in villages even today every morning you will find that. The ladies spray water and use rice flower to draw beautiful patterns, in front of the house that is taken as a daily routine. And those patterns, can be very easily generated using the L-systems using B-spline interpolation and just one or two rules.

(Refer Slide Time: 46:52)


Kolam patterns generated by L - system

For example, this is a kolam pattern this is called candies or (()). Actually the first you will get only this one this pattern it will there should be some more dots also which are not marked here actually. This is the first one and then when you combine four of them you will get this picture. Then if you combine four of such pictures you will get this again four of this can be combined to form a bigger pattern.

(Refer Slide Time: 47:31)


Candies

The above figure shows the kolam, pattern 'Candies' which can be generated by the following L-system.

Axiom : $(-D--D)$

Productions :

$A \to f++ffff--f--ffff++f++ffff--f$

$B \to f--ffff++f++ffff--f--ffff++f$

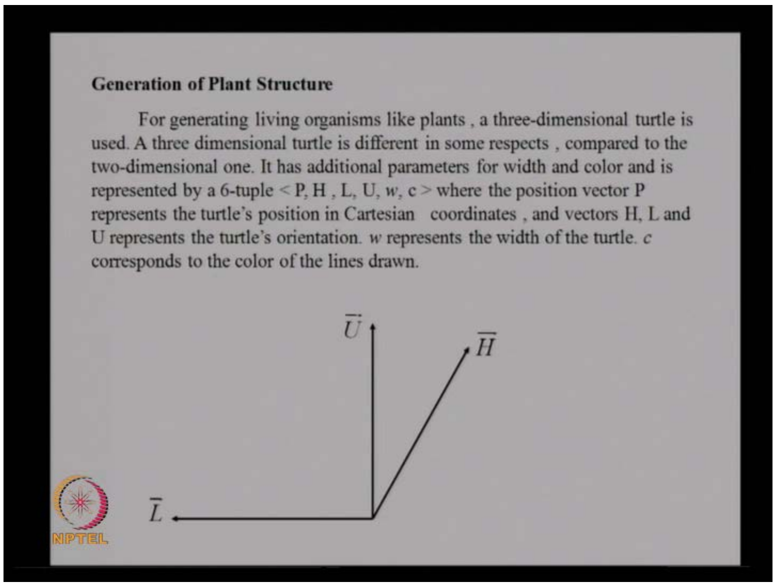
$C \to BfA--BfA$

$D \to CfC--CfC$

Angle increment $= 45^o$

So, starting with one axiom, which is this and a rule of this form four rules are there. Rule for D is this then for C is this A and B will be replaced by such strings. And in the next step again you will use similar patterns, you find that if you specify the number of iterations you will get beautiful patterns in this manner. Of course, successive points are joined by smooth curve using B-spline interpolation.

(Refer Slide Time: 48:14)

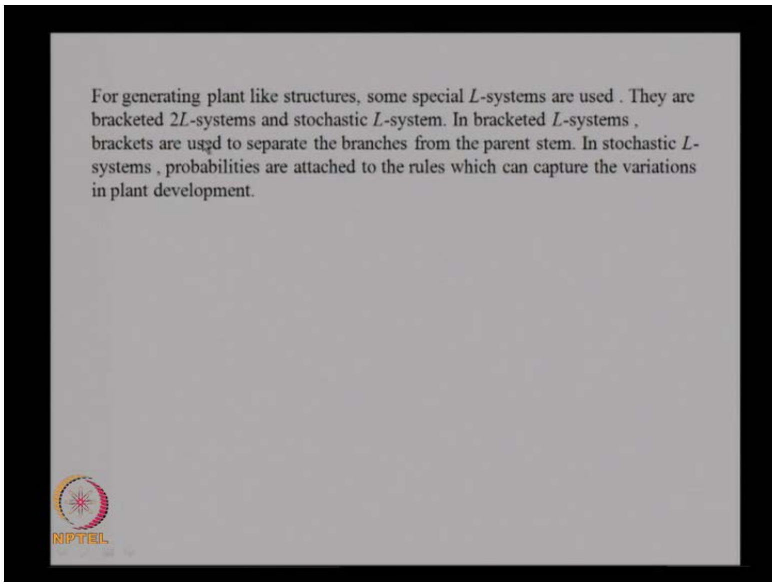


Generation of Plant Structure

For generating living organisms like plants , a three-dimensional turtle is used. A three dimensional turtle is different in some respects , compared to the two-dimensional one. It has additional parameters for width and color and is represented by a 6-tuple < P, H , L, U, *w*, c > where the position vector P represents the turtle's position in Cartesian coordinates , and vectors H, L and U represents the turtle's orientation. *w* represents the width of the turtle. *c* corresponds to the color of the lines drawn.

Now, plant structures can be generated and you can get beautiful imagery computer imagery using this. For generating living organism like plants, the three-dimensional turtle is used a three dimensional turtle is different in some aspects. Compare to the two dimensional one in the two dimensional one you have the x coordinate y coordinate and the angle orientation. Here you have the y z coordinates denoted by P that is a position vector which gives you the x y z coordinates. H L U, H L U give the orientation organelles the turtle. Then w represent the width and you can keep on changing the colour also, c represents the colour in which the lines have to be drawn. So, the three dimensional turtle has other parameters like with end colour.

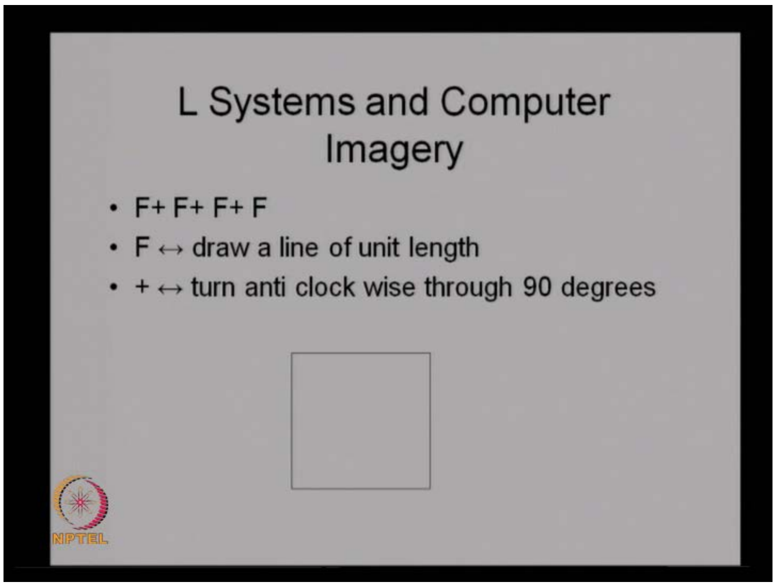
(Refer Slide Time: 49:32)


For generating plant like structures, some special L-systems are used . They are bracketed 2L-systems and stochastic L-system. In bracketed L-systems , brackets are used to separate the branches from the parent stem. In stochastic L-systems , probabilities are attached to the rules which can capture the variations in plant development.

So, for generating plant like structures some special systems are used making use if a three dimensional turtle. The special systems are stochastic L-systems, so with each rule you attach a probability see the growth if the plant is not identical. Suppose you sow six seeds all the six plants will not grow in the same manner. There will be slight difference and that sort of a thing is captured using probability. You attach some probability with the rules and that is called a stochastic system. Brackets are used to separate the branches that we have already seen you use brackets so that the branches are separated.

And sometimes, you use a different type of a bracket so that you want to move to something and draw then come back to the same position. Without drawing anything you use a different type of a bracket. Then in stochastic L-systems as I told you probabilities are used.

(Refer Slide Time: 50:39)

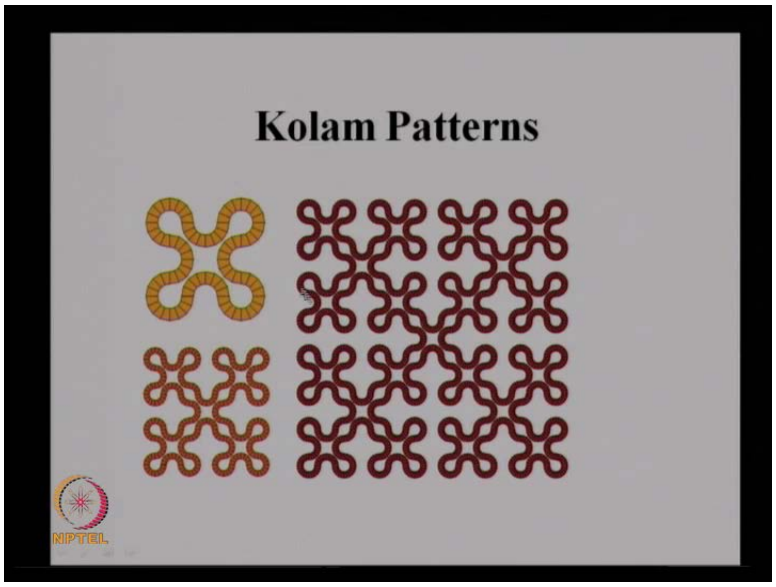


L Systems and Computer Imagery

- F+ F+ F+ F
- F ↔ draw a line of unit length
- + ↔ turn anti clock wise through 90 degrees

Also, cutting and then cutting a branch and when you cut how the branch grows all these are captured in computer imagery when you want to describe plants. Another example is this again f plus f plus f denotes this.

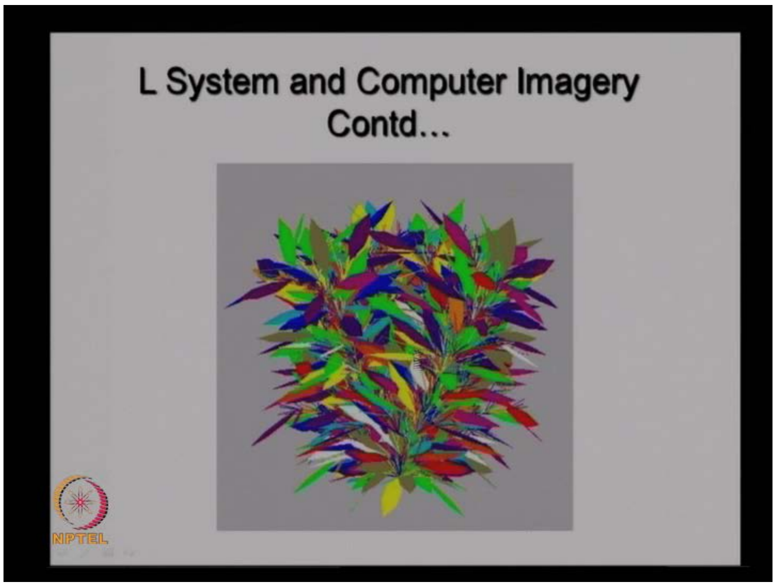
(Refer Slide Time: 50:58)



Kolam Patterns

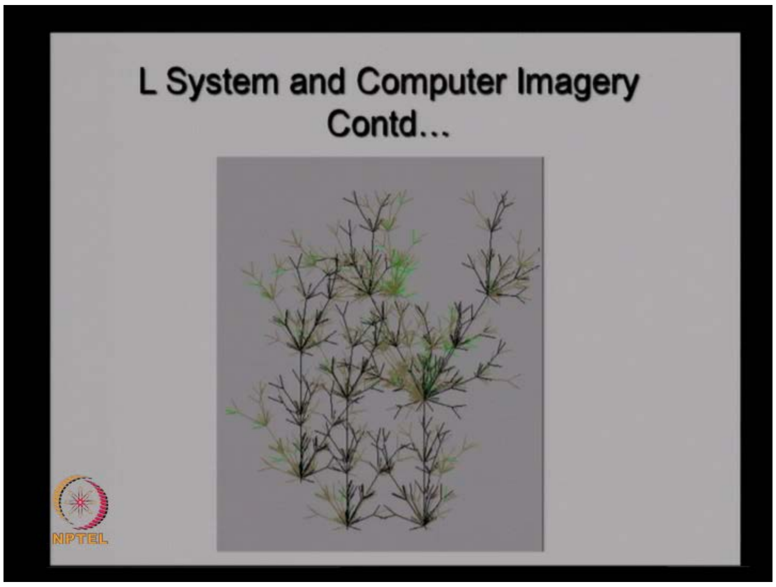
Then you use a certain rule you get another kolam pattern like this which is called a snake pattern, snake kolam. See that first one if you use spline approximation you may get this and then the second iteration when you use you may get this one kept four times is this.

(Refer Slide Time: 51:20)


L System and Computer Imagery Contd…

And then beautiful patterns like this can be generated.

(Refer Slide Time: 51:27)


L System and Computer Imagery Contd…

Some leaf pattern the growth of a plant like this with stems and so on. Here the same plant may not grow in a similar manner. So, if you use probability different types of plants can be got. And also you can have by cutting that this string this is represented as a string. And then the symbols in the strings are interpreted as move of a curser so if you cut this remove a portion of this string. And then use the growth the plant may grow in a different manner that can also be captured using L-systems. So, we defined that L-system is a different model from formal language theory point of view. Then main difference is parallel rewriting of symbols in every step.

And then you can interpret the symbols in a string and use it for computer imagery. In two dimensions you can have fractals generated you can have kolam patterns generated. And in three dimensions you can have plants and other structures generated using this system. Not only this even music can be generated using this in fact we have tried to use this to generate a some music. You write some sort of a tables, for each symbol there are seven swaras sa re ga ma pa da ni.

And for each one you can write a a rule in a similar manner and have tables. And apply these tables in some particular order and use a sound generating device along with that you can generate some particular raga some sort of a tune in that raga. Thus L-systems can also be used for generating some music. Not necessarily voice or anything, but some sort of a instrumental type of a music can be generated using this. And theoretically it is a different form always it is a a different form and then formal language theory point of view.

The study has been interesting some of the problems related to this that is given two deterministic zero L-system. Whether they are growth equivalent whether they represent the same growth was remaining as a whole as an open problem for a longtime and finally, it was solved.So, this is a another topic which is of interest. And recently there is something like watson greek zero L-systems that is also being explode. So, this is another advance topic which we are considering.