

Theory of Computation
Prof. Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute Of Technology, Madras

Lecture No. # 25
Problems and Solutions

(Refer Slide Time: 00:16)

Firing squad synchronization problem

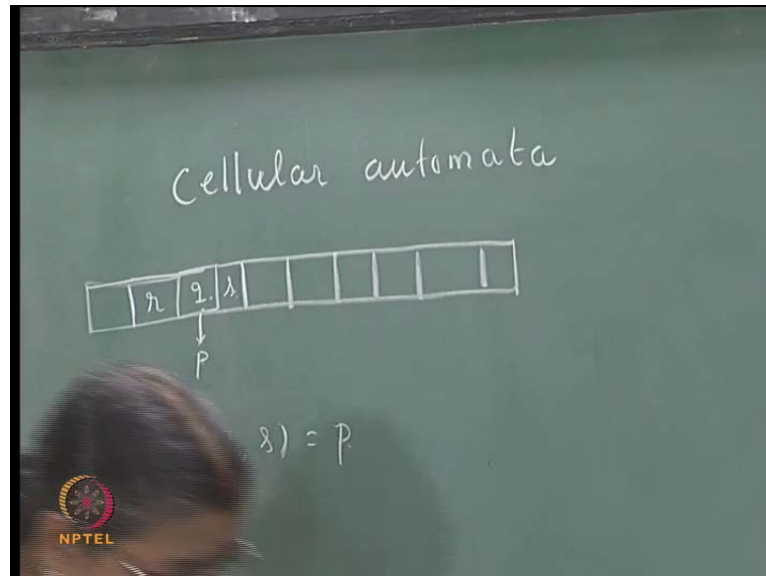
Consider a finite (but arbitrarily long) one-dimensional array of finite state machines, all of which are alike except the ones at each end. The machines are called soldiers, and one of the end machines is called a General. The machines are synchronous, and the state of each machine at time $t+1$ depends on the state of itself and of its two neighbors at time t . The problem is to specify the states and transitions of the soldiers in such a way that the General can cause them to go into one particular terminal state (i.e., they fire their guns) all at exactly the same time. At the beginning (i.e., $t=0$) all the soldiers are assumed to be in a single state, the quiescent state, when the General undergoes the transition into the state labeled 'fire when ready,' he does not take any initiative afterwards, and the rest is up to the soldiers. The signal can propagate down the line no faster than one soldier per unit of time, and their problem is how to get all coordinated and in rhythm.



Today, we shall discuss some problems. One is called the firing squad synchronization problem. What is this? I will read it out, this is called firing squad synchronization problem. Consider a finite 1 dimensional array of finite state machines all of which are alike except the ones at the end. The machines are called soldiers and the one at the one of the end machine is called the General. The machines are synchronous and the state of each machine at time t plus 1 depends on the state of itself and its 2 neighbours at time t . The problem is to specify the states in the transition of the soldiers in such a way that the general can cause them to go into a particular terminal state, firing state exactly at the same time. At the beginning all the soldiers are assumed to be in a single state, the quiescent state, when the general undergoes transition into the state labelled fire and ready, he does not take any initiative afterwards and the rest is up to the soldiers. The

signal can propagate down the line no faster than 1 soldier unit of time, and the problem is how to get all the coordinated and in rhythm.

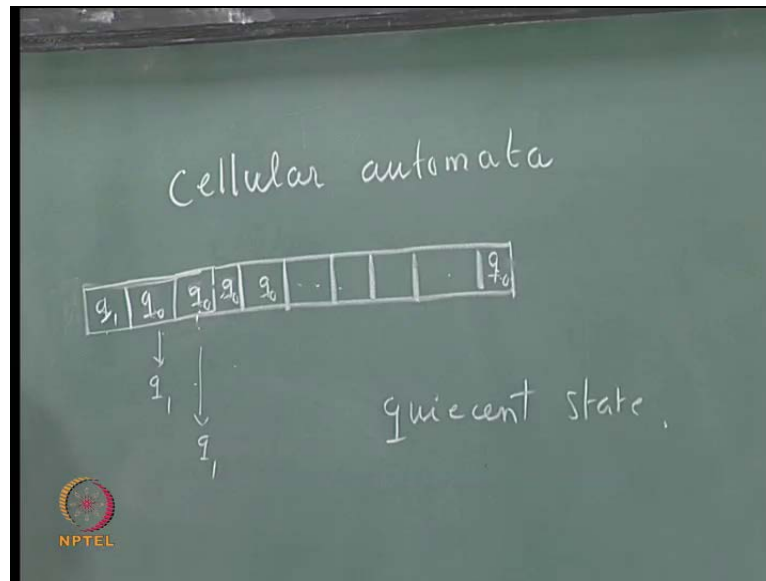
(Refer Slide Time: 01:37)



Actually, this is called a cellular automata, somewhat like a finite state machine you have 1 dimensional tape. I mean this, it is divided into cells. Each cell is in a particular state and the next instance depending upon each states, the state of its left neighbour and the right neighbour this will go to a state b. Suppose, this is in state r and s the mapping will be delta of r, q, s is equal to p.

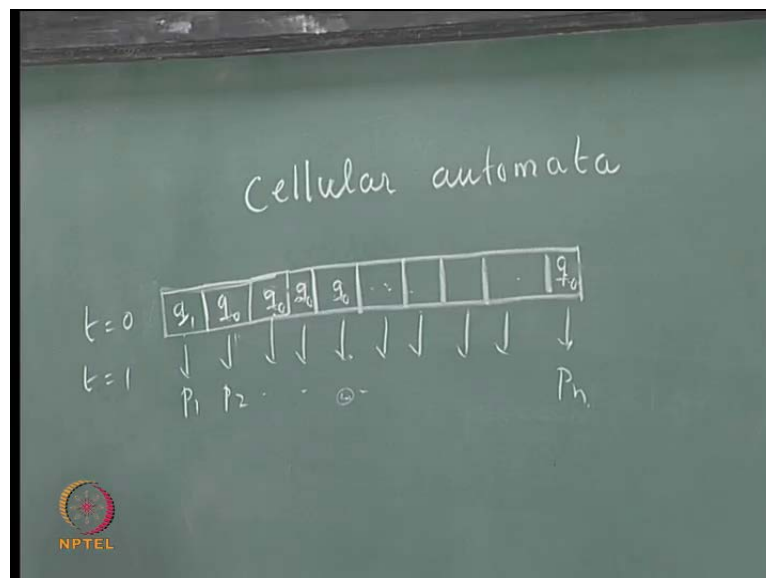
So, depending on its state and the state of the left neighbour and the state of the right neighbour, this will change from q to p. This is called a cellular automaton because you know that the, for this cell there is no left neighbour. We can assume something like this and the mu of this will be always in that particular state. So, this the mu of this depends only on it and the right neighbour. Similarly, the mu of this depends only on the left neighbour and its circle. Does not depend on the right neighbour, the last two cells are slightly different.

(Refer Slide Time: 03:13)



Now, all of them all these are all called soldiers and further. They are in a particular state called the quiescent state, that q naught is called a quiescent state. And general goes to a different state, the left most person is the general. So, he propagates some signal to the soldiers and when he propagates that signal the next step, depending upon the q_1 q naught and q_1 this may go to say q_1 and depending upon q_1 q naught and q naught this may go to q_1 in the next instance and so on.

(Refer Slide Time: 04:11)



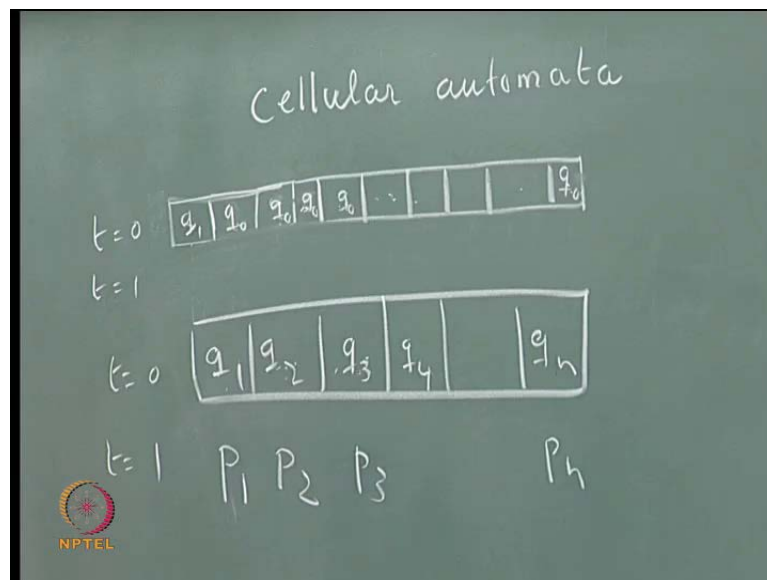
All these cells change state simultaneously. The next instance everybody will change the state, this is time t is equal to 0, at time t is equal to 1 everybody will change a state. So, next instance they will be say p_1, p_2, p_n the state of any particular cell was changed because of this left neighbour and right neighbour depending upon its own state left neighbour right neighbour it changes state. So, all of them change state simultaneously.

Madam it is the transition sequential $((\))$ balance

The transition is $((\))$.

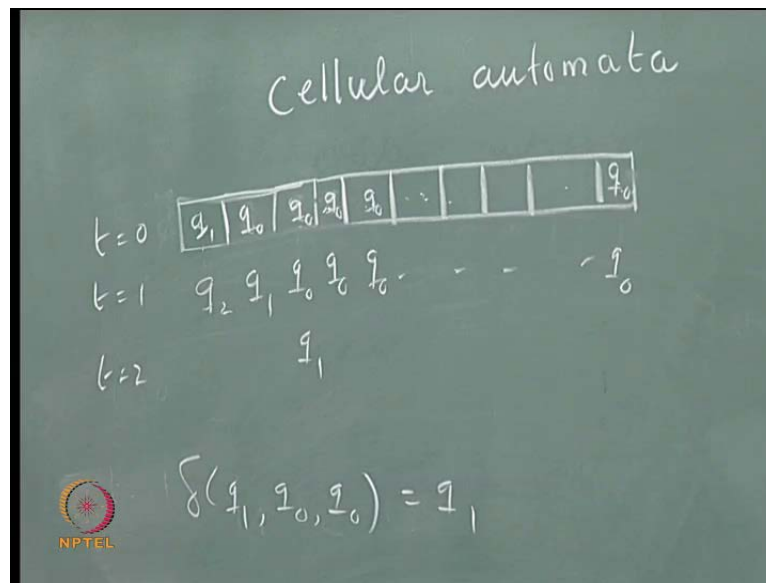
For the second cell is it delta of q naught, q naught, q naught or delta of let say p_2 q naught, q naught.

(Refer Slide Time: 05:09)



No, I will instead of q naught initially let me take it like this; q_1, q_2, q_n, n cells are in these states. Next instance q_1 will change it to p_1 depending on q_1 and q_2 , this will not consider left neighbour. Then this is t is equal to 0, t is equal to 1, q_2 will change it to p_2 depending upon q_2, q_1, q_3 . Then q_3 will change state depending upon q_2, q_4 and q_3 it will go to p_3 . So, this is the set of states at t is equal to 1, the next instance it will again change like that. All of them change state synchronously, parallelly.

(Refer Slide Time: 06:08)



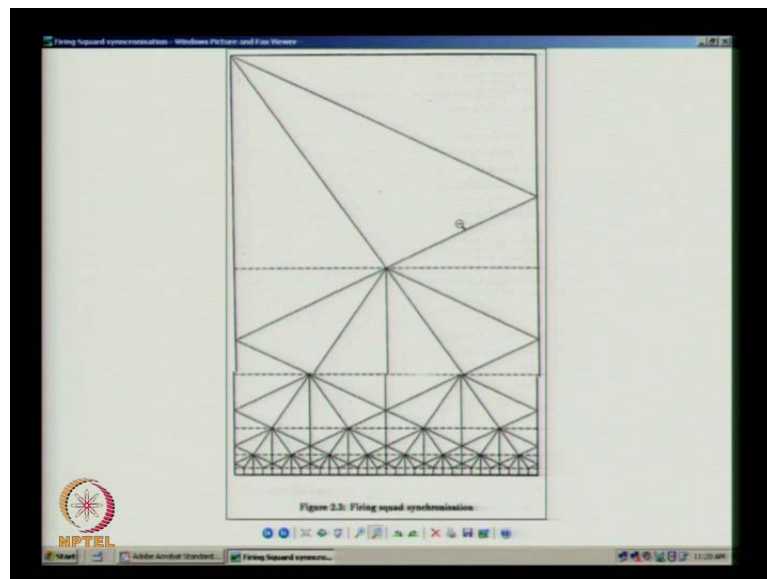
Now, what I meant by saying this is, initially all of them are in q naught. So, if a cell is in q naught the left neighbour is in q naught, right neighbour is in q naught this is called the quiescent state it will go to q naught only. So, next instance what happens is all of them will go to q naught, but this may go to say some q_1 or q_2 whatever it is you have to write the mappings and this will also change go to some state q_2 . So, next instance this t is equal to 2, this can go to q_1 . So, the mapping will be $\delta(q_1, q_0, q_0) = q_1$ it will be like that. Like that this q_1 state propagates like this. It can pass some signals like that.

So, at time t is equal to 2 there will be some sequence, t is equal to 3 there will be some sequence and so on. So, all of them should go to a firing state at a particular instance and none of them should go to that state earlier than that. None of the soldier should go to the firing state the earlier than that, all the soldiers general everybody will go to the firing state simultaneously. The order is given and they after sometime they have to fire simultaneously, this is called firing squad synchronisation problem. This was stated I think in 1957 by Myhill and lot of during that 1960s and the 1950s there was some discussion on that.

Now, the mapping or the delta mapping you have to write for that and that should not really depend on this n . This is the number of cells, the delta mapping which you are going to write should not depend on n . And one another thing is, all these are similar. If

you have delta mapping for this cell, you have the same delta mapping for this cell, you have the same delta mapping for this cell. Only the end cells this one will not depend on the left neighbour and this one will not depend on the right neighbour. So, rest of it every cell has the. So, if I have 20 soldiers or if I have 30 soldiers the mapping I should write is the same. It should not vary. So, write down such a mapping in such a way that all of them reach the firing state at the same time, but none of them reach the firing state earlier than that.

(Refer Slide Time: 09:17)



So, actually the mapping will be a lengthy process to write. The idea the solution for this is like this, initially this general he sends 1 signal like this, 1 signal like this. This is 1 signal, he is sending another signal which is 3 times slower or this is 3 times faster than this. So, this can be propagate 1 cell at a time, this propagates oneself in 3 units of time, this is 3 times slower than this and this signal when it reaches the last person it is reflected. So, you have to that reflection passing signal you have to take care in the states. So, they will meet here, when they meet it is the middle part. It will be 2 cells or 1 cell depending upon whether you are having even number of soldiers or odd number of soldiers and so on.

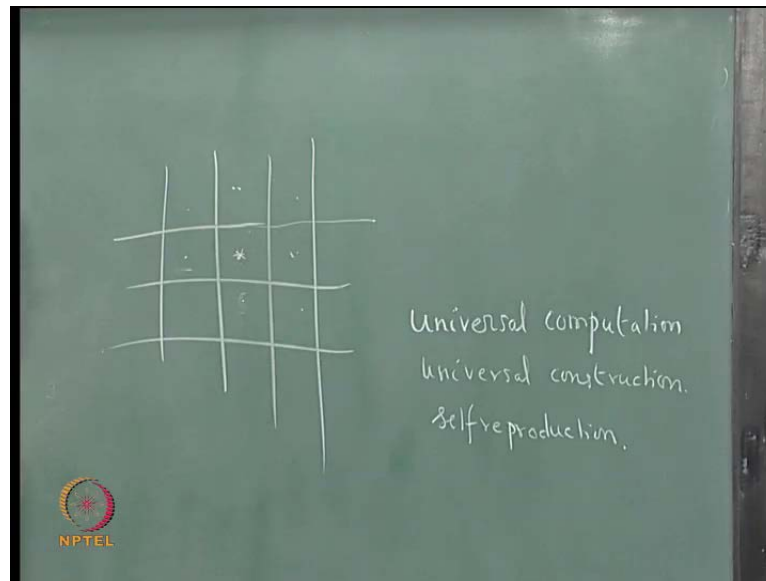
So, afterwards from this actually you can write 11, 12 or something like that. If you carefully write you can write down the mapping within that, the mappings do not depend on n the number of soldiers, but the time at which they fire depends on n . When do they

go to the firing state, this is the last is the firing state all of them go to the firing state that depends on n .

So, from here in the middle point 1 signal is sent here, another signal is sent which is 3 times lower. This signal when it reaches here gets reflected and when they meet it will be the midpoint of this portion. This is marked this some signal is sent like this. So, that this portion is marked and in this direction also 1 signal is sent like this and another signal is sent which is 3 times lower when this reaches the right end it gets reflected and reaches here. So, this middle portion is marked and now these 2 cells are marked, again they do the same thing. So, they send 1 signal here, 1 signal here and 2 signals which are slower 3 times slower like this, which is like this, which is like this and so on. So, in the middle point they will meet. So, all the middle points are marked then the next day all the other middle points are marked and so on. Then all of them will go to some star state simultaneously then that star when all of them are in star state they will fire.

So, you have to write down the mapping I mean this is a sort of a lengthy process, sit and write down, the idea is this. How much time it will take to go to the firing state time t is equal to t_0 you are starting, at what time you will reach the firing state you can give many solutions. In solution is not a unique one each one can give a different solution, it is not very difficult to give a solution which takes time $3n$ to $8n$ you can give something $3n$, $4n$. In between $3n$ and $8n$ you can give, but the optimum time is $2n - 2$, it has been proved that you can do this in the optimum time $2n - 2$ and for that the number of state required for a cell the delta mapping it will be the order of 1000.

(Refer Slide Time: 13:54)

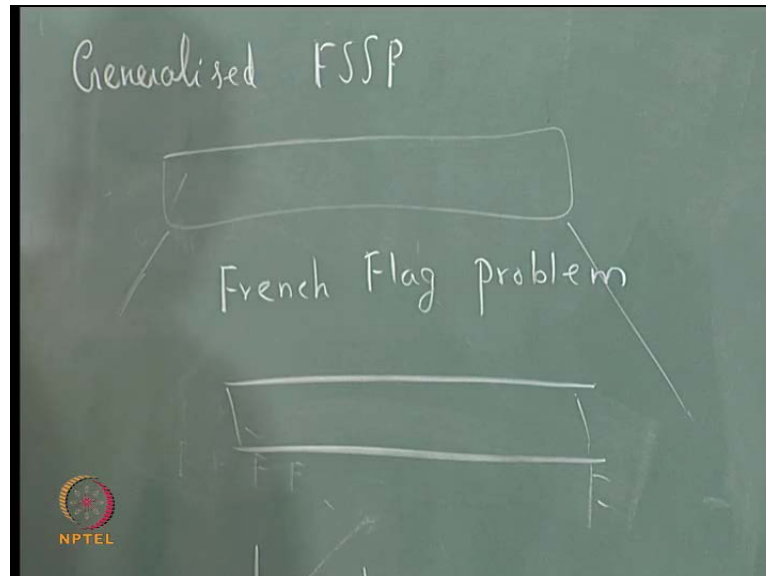


So, it is a very difficult a proof, but it can be done in $2n - 2$ moves, but something like $3n$ or $4n$ or $5n$ we can very easily give with say between 10 and 20 states. This was has been consider for a long time and having discussed this I will also mention what is meant by a cellular automata. This is called a cellular automata. You also have 2 dimensional cellular automata, you have cells, on each cell you have a final state automaton or this is a cell, this will change its state each 1, each cell will be in a particular state and this will change its state depending upon left neighbour, right neighbour, up neighbour, down neighbour depend and its own.

So, depending upon these 5 states it will change state. Now, such a cellular automata was first defined by John von Neumann. Von Neumann defined the cellular automata first. He has a book written self producing automata, self reproducing automata. Then they talk about universal computation, universal construction. Self reproduction, these things are talked about in cellular automata. Universal computation is like a universal tuning machine can compute any computable function. How can you do with the cellular automata, universal construction is given some basic portion, non blank portion here you can construct a mu of a tuning machine or something like that. I mean I will just put it like that you can you will be able to construct the encoding of a tuning machine or cellular automata for that matter. Self reproduction is given something it will make a copy of itself after some time and leave it to work by itself that is called self reproduction.

So, all these things were considered. They also talk about 8 neighbours, this is 4 neighbour, you can also have 8 neighbours and so on.

(Refer Slide Time: 17:01)



Lot of work has been done on cellular automata, this is called 1 dimensional cellular automata. Lot of work has been done on this and still work is going on and cellular automata have lot of application v l s i design (()) and v l s i test pattern then generation and all that it has got lot of application. Even in some physics and all it has applications because many of the cellular automata papers they appear in physical (()) and things like that. Another thing is called a generalised firing squad. Generalised firing squad synchronisation problem; what is this? This cell, number of cells when you are doing, this is also allowed to grow. When you allowed to grow, all of them change state at the same time 1 cell can split into 2 or 3 and that sort such a thing can also happen. That is called generalised firing squad synchronisation problem, this finds application in some modelling of biology.

In fact, they show that this model is useful when they want to observe the pattern on a tortoise, on the shell of a tortoise the things keep on changing, growing and at a particular time the pattern on the shell will suddenly change. So, they all reach the firing state or something like that they say. There is also something called French flag problem. We can say Indian flag also because Indian flag also has 3 colours. So, whatever you start with, these cells you start in any stage and finally, it will divide it may pass some

signals and finally, it will divide into 3 portions and say red, white, green something like that. It will become equally divide into 3 portions and all of them will be in the same colour.

So, that sort of thing is also possible. These are some these are really useful in L systems named after Lyndon Mayer which is useful in modelling plants and getting some beautiful pictures.

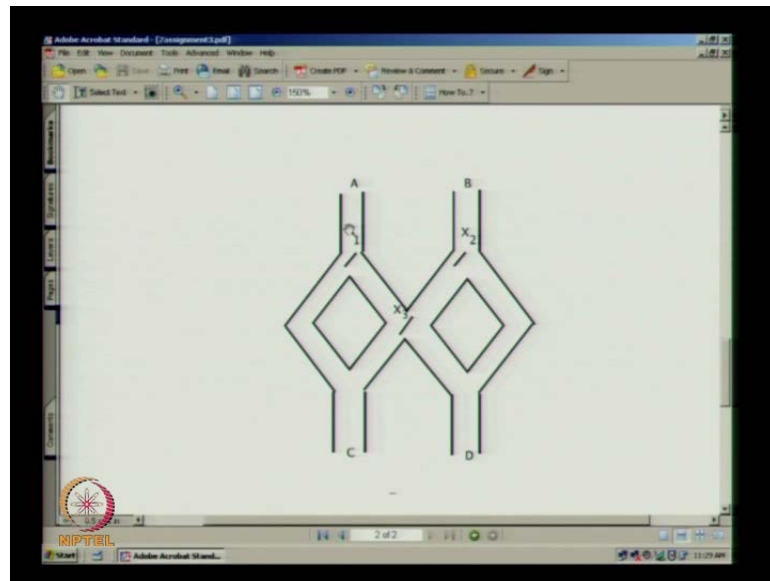
(Refer Slide Time: 19:08)

Consider the toy shown in the Figure. A marble is dropped in at A or B. Levers x_1 , x_2 and x_3 cause the marble to fall either to the left or to the right. Whenever a marble encounters a lever, it causes the lever to change state, so that the next marble to encounter the lever will take the opposite branch.



So, the actual mappings you have to write down now. The second question is consider the toy shown in figure 1, there are 3 rivers x_1 , x_2 , x_3 and a marble can be dropped in this place a or b.

(Refer Slide Time: 19:20)



When it is dropped in this hole it is considered as a 0 input, when it is dropped here it is considered as a 1 input and if it comes out of this hole, it is the output is taken as c and if it comes out of this hole the output is taken as d. Marble is dropped at a or b levers x_1 , x_2 cause the marble to fall either to the left or to the right. Whenever a marble encounters a lever it causes the lever to change state.

(Refer Slide Time: 19:46)


Consider the toy shown in the Figure. A marble is dropped in at A or B. Levers x_1 , x_2 and x_3 cause the marble to fall either to the left or to the right. Whenever a marble encounters a lever, it causes the lever to change state, so that the next marble to encounter the lever will take the opposite branch.



So that the next marble to encounter the lever will take the opposite branch; model this toy by a finite state automaton, denote the marble in a by 0 input and a marble in d by 1 input.

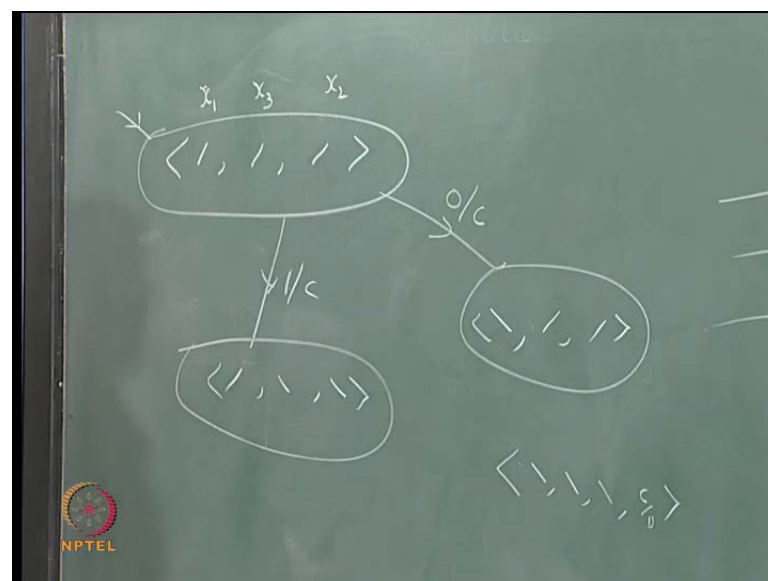
(Refer Slide Time: 20:02)

- Model this toy by finite automaton. Denote a marble in at A by a 0-input and a marble in at B by a 1-input. A sequence of inputs is accepted if the last marble comes out of D.
- Describe the set accepted by the finite automation
- Model the toy as a Mealy machine whose output is the sequence of C's and D's out of which successive marbles



Sequence of inputs is accepted if the last marble comes out of, comes out at the ocean d. Describe the set accepted model the toy as a mealy machine whose output is c and d.

(Refer Slide Time: 20:41)



So, how will you model this? Again the, writing the whole thing is a lengthy process the idea is like this. The state of the machine can be specified by a 3 tuple like this, like this

or something like a state can be specified like this. Denoting the position of the levers; this is say lever 1, we called it as $s \times 1, x 2, x 3$. I will call them as $x 1$'s position $x 3, x 2$ middle 1 is $x 3$. So, I will call it as $x 1, x 3$ either way you can change the order, does not matter. So, how many possible states can be, each one can be this or that. So, 8 possible states will be there. There will be 8, initial id is initial position in which suppose it starts in this initial position, this will be the initial state. Now, in this position you may get a marble through hole a or hole b. If you get a marble through a, what will the direction, what will be the direction it will take? It will fall like this and it will go here.

So, output will be c and it will change the possession of this, but these 2 will be unaffected. So, if you get a 0 the output will be c and you will go to the state, the first 1 will change into this, second and third are unaffected. Now, if you get a 1 or if the marble falls through the b hole what will happen, see this is like this. So, the marble has to come like this, like this and it will go like this. So, it will affect the second 1 and the third 1, the first 1 will remain as it is and the output will be a c. So, like this you can model the whole thing as a mealy machine or a as a mealy machine.

Now, when I want to consider this as a input output devices, this is ok. When, I want to consider it as a accepting device I have to also look into the output. So, actually the state should be taken as a fourth topple, this will be c or d, the last output is c you go to whatever it is $x 1, x 3, x 2, c$, if the last output is c, if it is c here you will go to this comma c the last output is d it will go to $x 1, x 3, x 2, d$.

(Refer Slide Time: 24:07)

- Model this toy by finite automaton. Denote a marble in at A by a 0-input and a marble in at B by a 1-input. A sequence of inputs is accepted if the last marble comes out of D.
- Describe the set accepted by the finite automation
- Model the toy as a Mealy machine whose output is the sequence of C's and D's out of which successive marbles



So, when d is there it will be a final state, the last output has to be a d. Condition, if you want to look at it as a input output device a sequence of this is accepted if the last marble comes out at d. So, now you have how many states? How many states you have? 16 states you have and if you have d in the last component that will be a final state, see initially there will not be any output, first output will not be there. Only, when it comes through it will be output. So, initially 1 more initial state you can have 16 plus 1, 17 you can have, does not matter. So, this is a way you model it the details have to be worked out. Set you have to solve by regular expression by solving. It is not a very or you have to look at the pattern, the pattern will not be very easy to look at the only way you can describe the set is by means of a regular expression for which you have to write down the equation and solve.

13, why?


After removing unreachable states

Unreachable states can be removed, you have to work out the details, unreachable states can be removed, if you remove them there will be there can be less number of states, by then you have to write down the equation for them and solve for the initial state which will give you the regular expression. That regular expression will give you the set of strings accepted. So, it is a lengthy process you have to do it, it is a method.

(Refer Slide Time: 25:47)

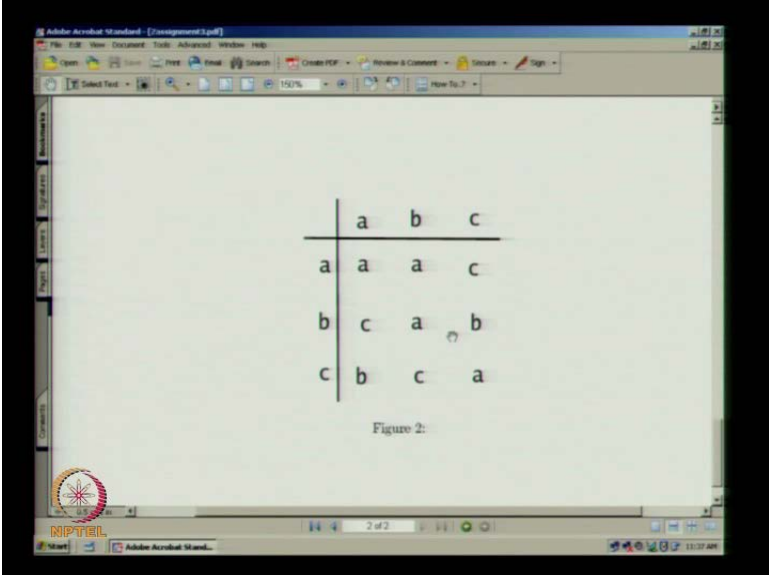
Design a DFA or an NFA for the following language.

- The set of all strings over the alphabet { a, b, c } that have the same value when evaluated left to right as right to left by multiplying according to the table given in the Figure.



So, going to the next problem give deterministic or non deterministic finite automaton for the following language, the set of all strings over the alphabet a b c that have the same value when evaluated from left to right and right to left by multiplying according to the following non associative table. What is the table? This is the table.

(Refer Slide Time: 26:11)



	a	b	c
a	a	a	c
b	c	a	b
c	b	c	a

Figure 2:

So, you give a final state automaton which will accept strings which give you the same value when evaluated from left to right or right to left, this table is a non associative table and a non commutative table.

(Refer Slide Time: 26:39)

The chalkboard displays three mathematical expressions illustrating the evaluation of a sequence of elements $a_1, a_2, a_3, a_4, \dots, a_n$.

$$\left(\left(\left(a_1 a_2 \right) a_3 \right) a_4 \right) \dots a_n$$
$$\left(\left(a_1 * a_2 \right) * a_3 \right) * a_4$$
$$a_1 * \left(\dots \left(a_{n-3} * \left(a_{n-2} * \left(a_{n-1} * a_n \right) \right) \right) \dots \right)$$

The NPTEL logo is visible in the bottom left corner of the chalkboard image.

So, I explained earlier what is meant by a right evaluation and left evaluation a_1, a_2, a_3, a_n the right, left evaluation is this, a_1, a_2 then this, then this and so on. That is you first considered $a_1 * a_2$, then $* a_3$, then $* a_4$ this is the left evaluation. What will be right evaluation, the right evaluation will be $a_{n-1} * a_n$ will be calculated first then a_{n-2} this then a_{n-3} and so on. Until you get $a_1 * a_n$ like this, it is not a n see the usual mistake people do is consider $a_n * a_{n-1} * a_{n-2}$ that is not correct, given this problem many make this mistake, this is not correct right evaluation means this.

(Refer Slide Time: 28:07)

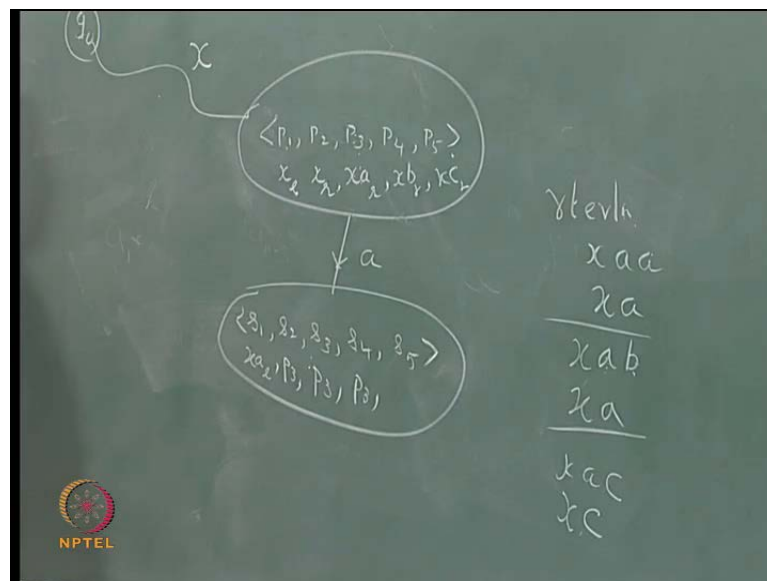
The chalkboard displays the sequence of elements a, b, c and their possible permutations:

$$a \ b \ c \quad a \ b \ c$$
$$a \ c \quad a \ b$$
$$c \quad a$$

The NPTEL logo is visible in the bottom left corner of the chalkboard image.

Now, what sort of strings will have the same value when evaluated from left to right or right to left for example, take 1 string a a a whether you evaluate a a gives only a. So, whether you evaluate from left to right or right to left it gives the same value. So, it has to be accepted. Now, suppose I have a b c what will happen? Let me check. What is a b? a b is a, a c is c, left evaluation is this c. What will be the right evaluation? Let me check. May be it also b c. What is b c b and a b is, a b is a, a b is a they are not the same.

(Refer Slide Time: 29:34)



So, a b c should not be accepted. Any string which gives you the same value when evaluated from left to right should be accepted, other thing should not be accepted. How will you describe this with a final state automaton? You can have a d f s a as follows, the construction is like this. The d f s a will have $3^5 + 1$ state. How is that? What is 3^5 ? $243 + 1$, 244 states. $81 + 1$, 243 plus 1, 244 states. 244 states it will have. How are the states defined initial state is there on q naught, apart from that there will be the states are denoted by 5 topples, 1, 2, 3, 4, 5 topples and each one can be a b or c, each of the component can be a b or c.

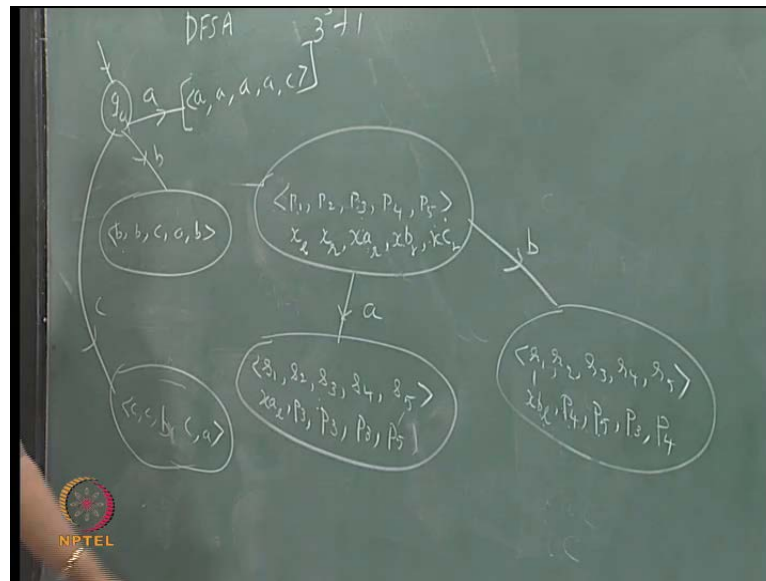
So, first component can be a b, a b or c second component can be a b or c. So, that is why we get 3^5 . So, each state it is denoted as a 5 topple, now what does that 5 topple mean? The meaning is like this, starting from the initial state after reading x you go to a state p 1, p 2, p 3, p 4, p 5 and what it means is p 1 is the left evaluation of it will denote the left evaluation of the string x. And p 2 will denote the right evaluation of that

string and p_3 will denote the right evaluation of $x a$, p_4 will denote the right evaluation of $x b$, p_5 will denote the right evaluation of $x c$.

So, these 5 letters $a b c$, $a b$ or c they will denote if you reach this state from q naught after reading a string x means the first component denotes the left evaluation of x , second component denotes the right evaluation of x , third component denotes the right evaluation of $x a$, fourth component denotes the right evaluation of $x b$, fifth component denotes the right evaluation of $x c$. So, after that how do you define the moves. So, if you have a 5 tuple like this, if you read a you will go to another state, which is say s_1, s_2, s_3, s_4, s_5 . Now what is s_1 ? s_1 is the left evaluation of $x a$, you know what is this? This is one of the $a b c$. So, from that and a by looking into the table you can fill this. You will be able to fill that, this is the left evaluation of $x a$.

Now, what is the second component s_2 , it is the right evaluation of $x a$, isn't it. From here after reading $x a$, you have reached this. So, second component should give you the right evaluation of $x a$, but what is the right evaluation of $x a$, it is p_3 . So, second component will be whatever value you have that will be here, now what about the third component? Third component is the right evaluation of right evaluation of $x a a$, but what is $a a$ from the table a . So, it is the right evaluation of just $x a$. So, s_3 will also be p_3 . Now what is s_4 ? It is the right evaluation of $x x a b$, it is the right evaluation of $x a b$. What is $a b$ from the table a . So, that is the right evaluation, right evaluation means first you have to evaluate this you know you know $a b$ is a . So, it is the right evaluation of $x a$, right evaluation of $x a$ is again p_3 . So, this will be p_3 , what is the last 1? It is the right evaluation of $x a c$ what is $a c$? $a c$ is c .

(Refer Slide Time: 29:34)



So, it is the right evaluation of $x c$. What is the right evaluation of $x c$ that is $p 5$. So, $s 5$ will be $p 5$. So, from one s topple after reading a to another s topple you will go and this s topples you can determine in this manner. Similarly for example, if you take $a b$ let me take $a b$ here then you will go to a s topple $r 1, r 2, r 3, r 4, r 5$. What is $r 1$? That is the left evaluation of $x b$ you know the value of left evaluation of $p 1$ multiply with b you will get the left evaluation. What is this one? This is the right evaluation of $x b$, right evaluation of $x b$. What is the right evaluation $p 4$?

So, you will get $p 4$ here and what is $s 3$? It is the right valuation of right evaluation of not (c) right evaluation of $x b a$. Up to this you have a read $x b$, then multiply by a . So, what is $b a$ from the table? $b a$ is c that is $x c$, $x c$ is $p 5$. So, this will be $p 5$ whatever you have, whatever value you have you will write that. Each one of them will be either $a b$ or c . Is that clear and what about $r 4$? It is the right evaluation of $x b b$. What is the right, what is the value of $b b$? $b b$ is a . So, that the right evaluation of $x a$ and what is the right evaluation of $x a$ here? $p 3$. So, this will be $p 3$. $r 4$ is nothing, but $p 3$ and what about $r 5$? It is the right evaluation of $x b c$. From the table what is $b c$? $b c$ is b . So, the right evaluation of $x b$, right evaluation of $x b$ is $p 4$. So, you will get $p 4$ here.


So, $r 1$ is the left evaluation of $x b 1$ which you have to determine then $r 2$ is just $p 4$, $r 3$ is $p 5$, $r 4$ is $p 3$, $r 5$ is $p 4$ from this. Similarly, for each state you have to write down the successor, a successor, b successor, you can write down all the mappings and initially

starting from q, q naught after reading a to which state you will go? You will go to a, left evaluation is a, right evaluation is a. And the right evaluation or left a a a, a b, a c that is a a c, a a c. After reading a b you will go to b b, what is b a? b b, b c, c a b, c a after reading a c you will go to c c, c a, c b, c c that is b c a. So, from one 5 topple to another 5 topple how you can go that we have already seen and what are the final states? The first and the second components are equal same, those states are marked as final state. If the first component and the second component if they are equal, those states are denoted as final state. Infact these are all final states, this will be a final states, this will be a final state, this will be a final state. So, this is a deterministic diagram and this will accept that particular set. This is not the only solution, you can give non deterministic solutions also there are other methods possible.

(Refer Slide Time: 40:18)

Let $\text{value}(x)$ be the result when the symbols of x are multiplied from left to right according to the table given in the Figure.

- Is $L_1 = \{ xy \mid |x| = |y| \text{ and } \text{value}(x) = \text{value}(y) \}$ regular?
- Is $L_2 = \{ xy \mid \text{value}(x) = \text{value}(y) \}$ regular?



So, it is not a sort of unique solution. It is one of the solution which gives you a deterministic machine. This again actually this is the second portion of your this is left evaluation, value of x is the result when the symbols are x or multiplied from left to right in the table as in the table that is you are considering left evaluation only now.

Now, you are defining a language L_1 set such that it accepts strings of the form xy where the string is of even length, x, y should be of even length and you divide it into half half and the valuation of the first half is the same as the valuation of the second half. The second one is you have the same thing, but now they are not dividing them into equal

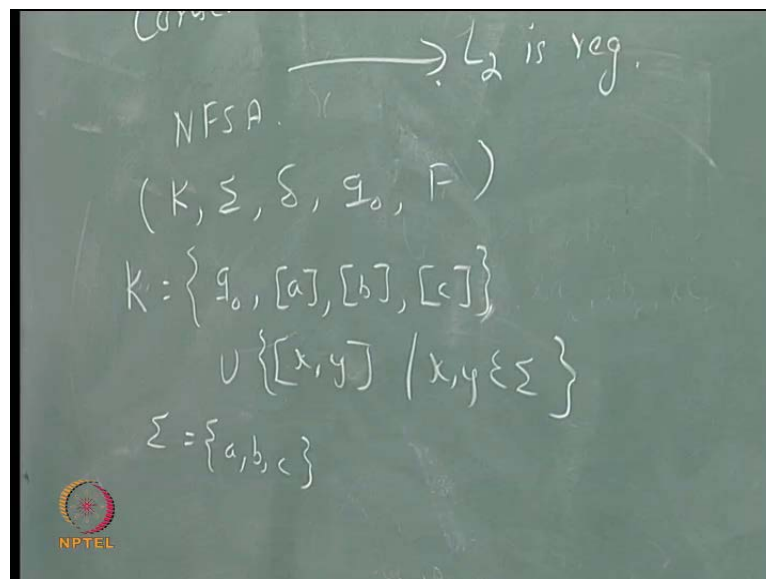
half's, you are splitting the string and the first portion gives you value of x again, second portion gives value of y if they are equal. You can split in any manner you want. So, the strings in the L 1 starting from the initial state you accept a string, if this can be divided into equal portions such that the value of this is equal to the value of this, in the second one L 2 you accept strings like this, where the split can be there need not be into equal portion, length of x need not be equal to length of y, you can split anywhere you want and the value of x should be equal to value. It should be possible to split it in such a way that the value of x is equal to value of y, x may be longer length, y may be longer does not matter. How many of you were able to do this is L 1 regular, is L 1 regular.

(())

is L 2 regular?

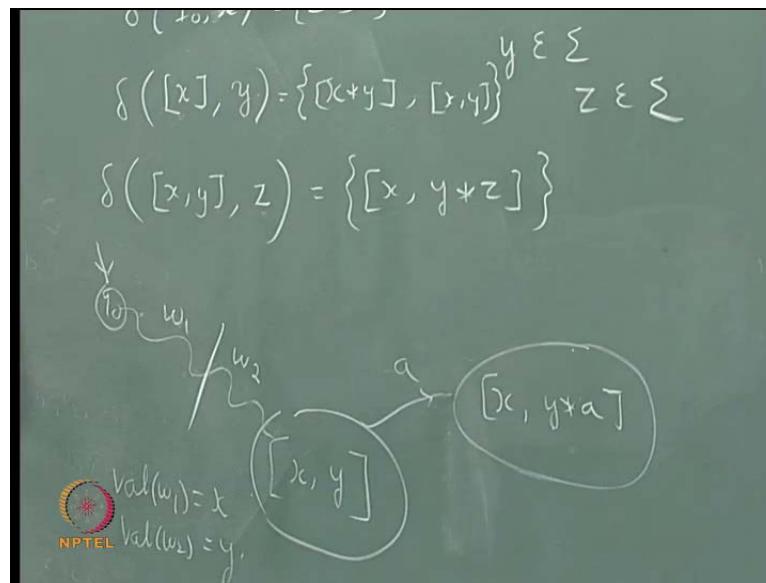
Yes

(Refer Slide Time: 42:36)



L 2 is regular, but L 1 is not regular. L 1 is not regular, L 2 is regular. So, how do you show that to show L 2 is regular you have to construct a final state automaton. For this construct F S A, you construct N F S A like this k sigma delta q naught f where k has states like this q naught, then the symbols are a b, you have a b c and also x y, x y because sigma is a b c, the states are initial state then single ton states, 3 states then pairs a a a, a b, a c b, a b b, b c, a c b, c c.

(Refer Slide Time: 44:48)

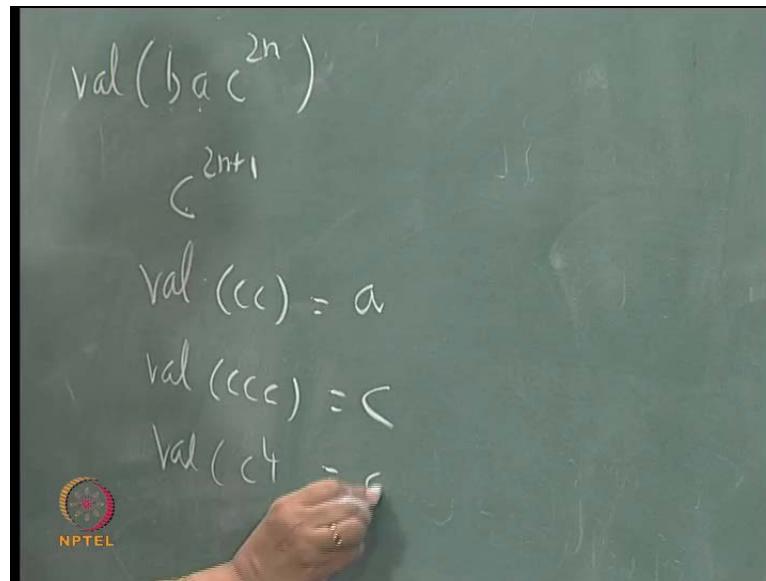


Now, we q naught is the initial state. f consists of a a, b b, c c. Now, you have to write the delta mappings. It is defined like this, delta of q naught a or I will write as q naught x is equal to x, x can be a b or c, x belongs to a b or c. Then from this state if you get a y then y is also symbol x is a state with this flower brackets what will be this? There are 2 possibilities here either you go to x star y or x comma y. This is the single state, this is pair.

Now, if you have a pair x y and z coming because x y z also belongs to sigma. What will be this? This will be x comma y star z. This is how the delta mapping is defined in a sense suppose in from q naught after reading a string you go to a pair x y, you go to a state like this. That means, it is possible to split it in such a way w 1, w 2 such that value of w 1 is equal to x, value of w 2 is equal to 1. So, starting from q naught after reading a string w 1, w 2 if you go to x y then it means that the string w is w 1, w 2 you are able to split it in a such a manner any where you can split such that the first portion evaluates x, second portion evaluates to y. That is why this if the left evaluation and the right evaluation are the same you accept. From this if I get a to what state do I go? I go to x y star a. So, this portion evaluates to x and w a, w 2 a evaluates to y star a. So, you are justified in that, writing this mapping and what about this mapping? When you are after reading a string w from q naught, if you are in you are in a state w the left the evaluation of left x this would mean the evaluation of w. I will write it as x where value of w is x

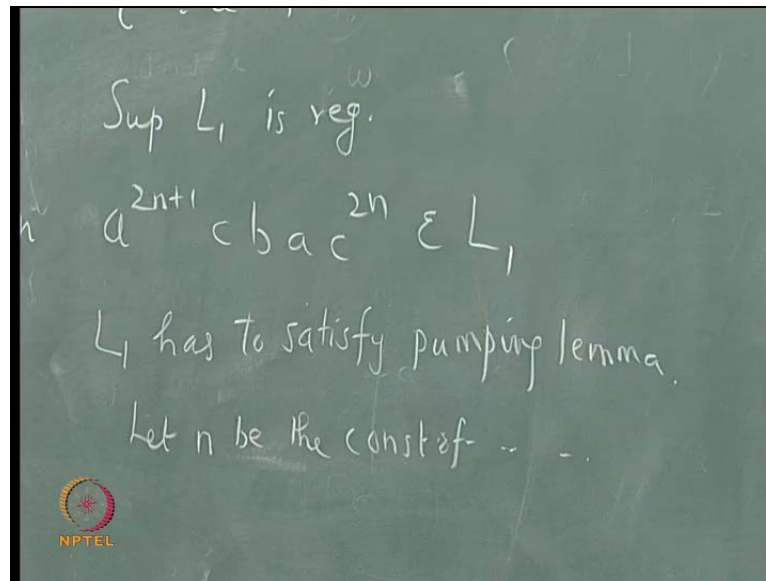
Now, after reading a you can read, go to x a, that is the value of x a is this that is again satisfied or you can go to, there are 2 possibilities this is non deterministic, x comma a. That is this w a you are able to split like this, the value of the first portion is x when you multiply, the second portion is a. So, you are justified in writing the mappings like that. So, this we have constructed this will accept exactly L 2.

(Refer Slide Time: 50:21)



So, L 2 is regular. Now, we have to show that L 1 is not regular, L 1 is not regular. Usually to show something is not regular what methods do you use? Something like pumping Myhill- Nerode theorem and so on. We will use pumping lemma here, consider this is not the only solution you may give a slightly different solution. I will give you 1 solution, a power 2 n plus 1 c b a c power 2 n. Look at this string, I have to divide into 2 half's, I will to divide it in a middle. What will be the value of a power 2 n plus 1 c, what will be this? a a a a will give you only a, a c will give you let me have the table. What will a c c. What is the value of b a c power 2 n? b a is c. So, this is c power 2 n plus 1, b a s c. So, you get c power 2 n plus 1, what is c c? What is the value of c c? a. What is the value of c c c? That is a c that is c. What is the value of c power 4? That is c a and no no c c which is a.

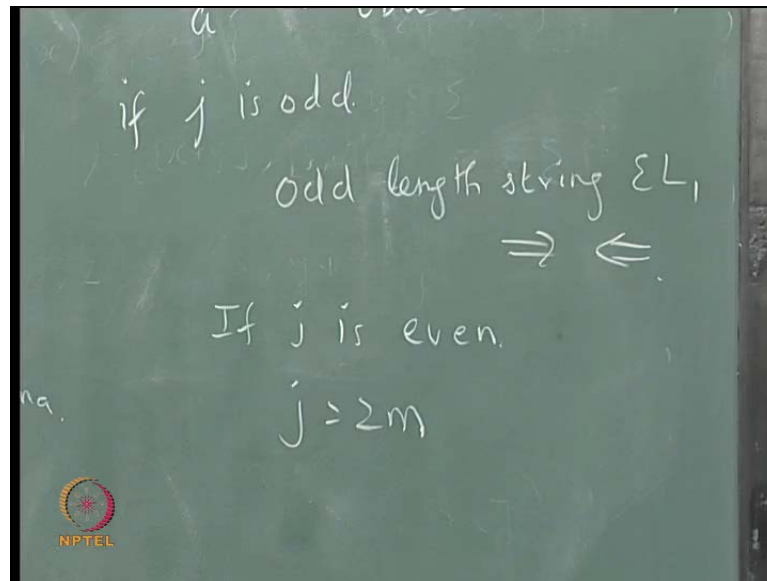
(Refer Slide Time: 52:26)



So, c power m equal to c if m is odd, c power m is equal to a if m is even. So, here you are having odd c , this is the same as value of this. So, this portion evaluates to c . So, this portion evaluates to c . What does that mean? This satisfies the condition given here. So, the argument has to start like this. Suppose L_1 is regular, suppose L_1 is regular then $a^{2n+1} c b a c^{2n}$ belongs to L_1 because it satisfies the condition. This is for any n . I have not taken any particular n this is for any n , for all n . Then L_1 has to satisfy the pumping lemma, L_1 has to satisfy pumping lemma and let L be the constant of the pumping lemma.

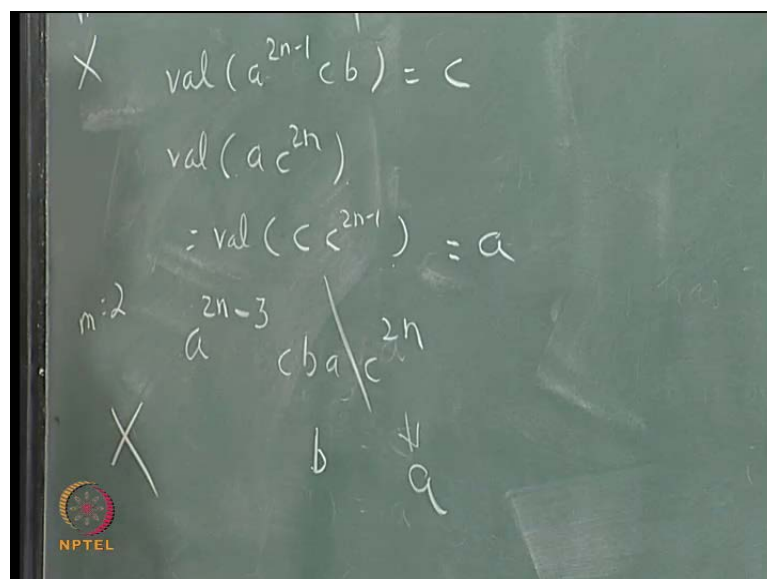
So, when you take this string if you apply the pumping lemma where will the pump occur? Pump occurs within this, within the first n first prefix of length n the pump should occur isn't it. So, pump will occur in a . So, suppose a^i is the pump, that is you can split this string in $u v w$ where v is of the form a^i , such that $u v^i w$ belongs to L for all i greater than or equal to 0 , i can be 0 also such that $u v^i w$ belongs to L for all i greater than or equal to 0 , this is the pumping lemma.

(Refer Slide Time: 55:17)



Now, if you do that a power $2n + 1$ minus i , take i is equal to the case when it is 0. So, I have used i , I will say j , the pump is a power j . So, this is j , here also I have used i , here also I have used i , 1 of them can be different just to make it not confusing $c b a$, $c b a$, c power $2n$ should belong to L_1 , if j is odd, if j is odd what do you get? The length of the string is odd by our condition the string has to be of even length then only you can split into 2 equal halves. So, you get an odd string belonging to L_1 which is a contradiction, odd length string belongs to L_1 and that is the contradiction. If j is even, we have to consider the both possibilities, j you can write as $2m$.

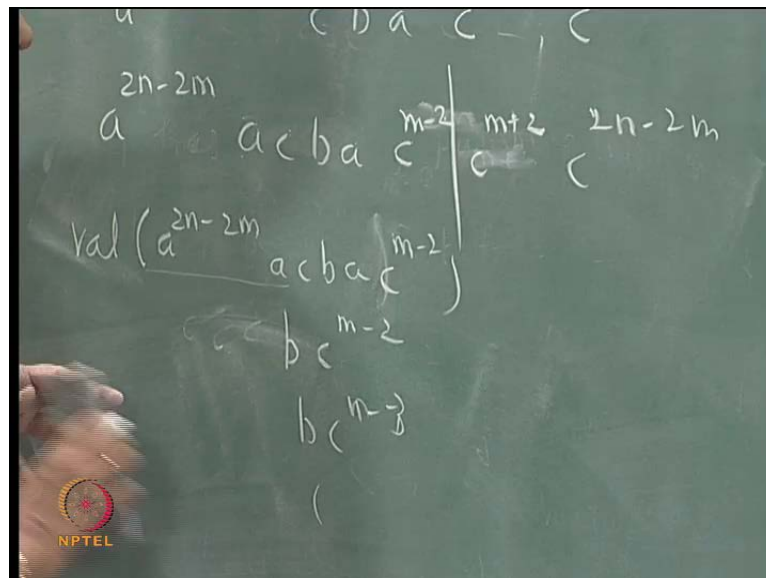
(Refer Slide Time: 57:09)



If you write like this, what happens? You come to the conclusion a power $2n + 1$ minus $2m$, $c b a, c$ power $2n$ should belong to L_1 , it should belong to L_1 . We will show that this cannot happen, we have to show that this cannot happen, odd, if j is odd we have already seen. Now, this you cannot do it in 1 step. You have to do consider several possibilities.

Suppose m is equal to 1, if m is equal to 1 what will be this? It will be a power $2n - 1$, $c b a, c$ power $2n$, then where will this split occur? Split occurs here, this string is of $2n + 1$, this is of $2n + 1$. So, what would be the value of this? What be the value of a $2n - 1$ $c b, a c$ is this will be a , then $a c$ will be c , $c b$ will be c . What will be the value of a c 2 power n that is equal to value of, what is a c, c power $2n - 1$ that is a, c power n is a if n is a 1, c power n is c if a it is r . So, they are not equal. So, m is 1 it does not work out, then m is equal to 2, when n is equal to 2 this will be a 2 power n plus 1 minus 4 that is minus 3, $c b a, c$ power $2n$. Where will the split occur? It will occur here, this is of length $2n$, this is of length $2n$. So, this will evaluate to a and what will be the evaluation of this? This is $a, a c$ is $c, c b$ is, $c b$ is $c, c a$ is b . They are not the same.

(Refer Slide Time: 00:01)



So, this also is ruled out. Then see the split, if it occurs within this portion you have to be careful. If the split is going to occur in the c we have to consider separately. So, m rest of the case m greater than 2. So, in that case the string is a 2 power n plus 1 minus $2m$, $c b a, c$ power $2n$ and this I can write as a power $2n - 2m + 1, c b a, c$ power, c

power $2n - 2m$, c^m or I can write it as $a^{2n - 2m} b^m$, $a^{2n - 2m} b^m$, $c^{2n - 2m}$. So, this portion is length $2n - 2m$, this $2n - 2m$, this one I have to split into equal halves. So, this itself I can write as $m - 2$, $m + 2$. So, what will happen? This is $4 + m - 2$ is $m + 2$. So, the split will occur here. What will be the length of this? $2n - 2m + m + 2$, this is $2n - m + 2$ split will occur like this.

Now, what will be the evaluation of this? Value of $a^{2n - 2m} b^m$, $a^{2n - 2m} b^m$, will evaluate a then a^c is c , c^b is b , b^a is c .

$c^{c^m - 2} c^c$, c^m

c^b is c^c a is (c)

Then $a^c b^a c^{m - 2}$. What is a^c ? c , what is c^b ? c , what is c^a ? b , what is b^c ? b . So, b^c power $m - 1$, b^c minus $m - 3$ and so on. Ultimately it will become b , this will become b and what about this portion, this is $c^{2n - m + 2}$ any string of c 's depending on m is even or odd this will be even or odd, the power will be even or odd, either way it is either it will evaluate to c or a . It will not evaluate to b . So, this is also not ruled out. So, whether j is odd or even the pumping lemma shows that it is not regular. So, we will stop at this.