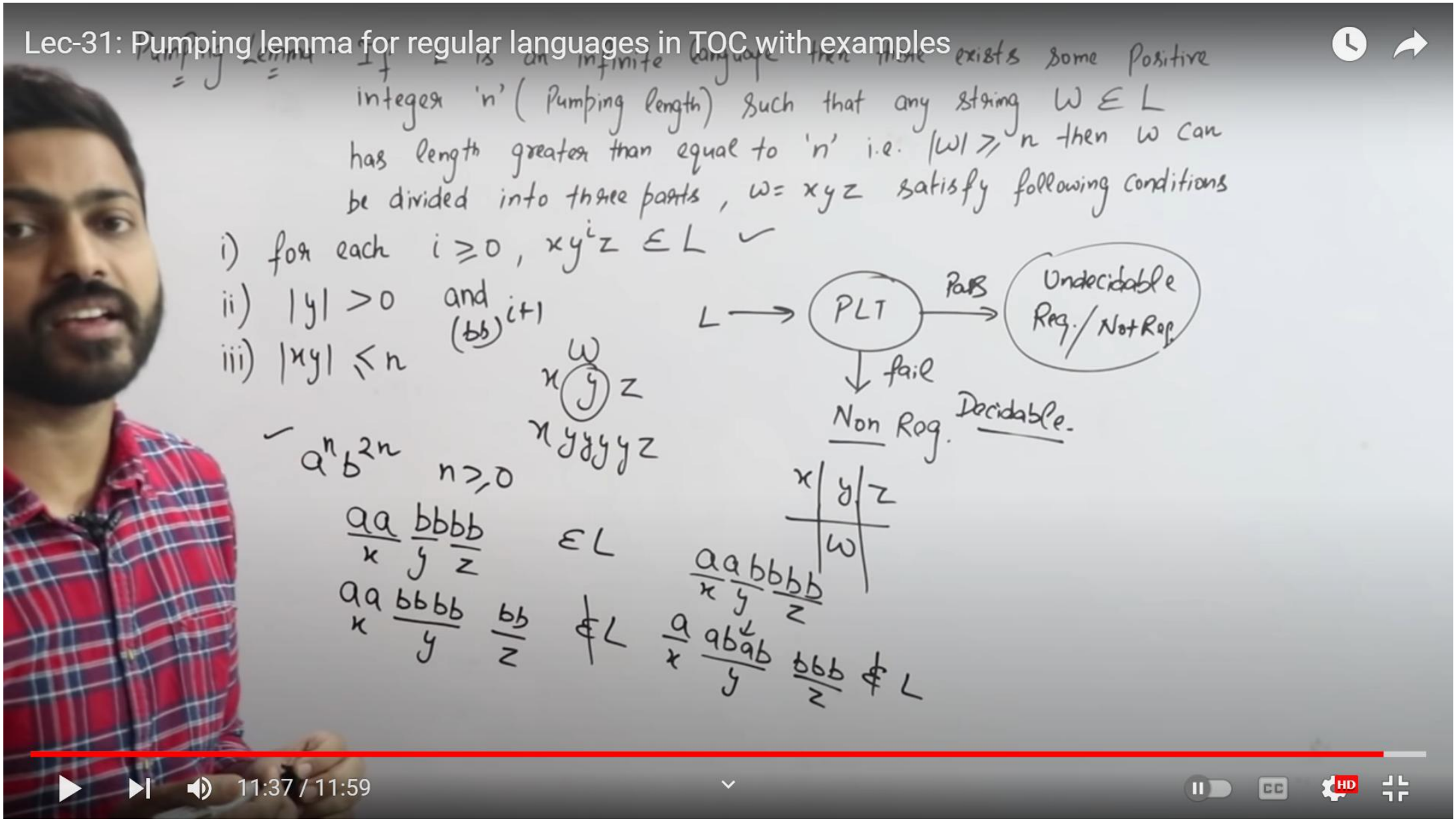
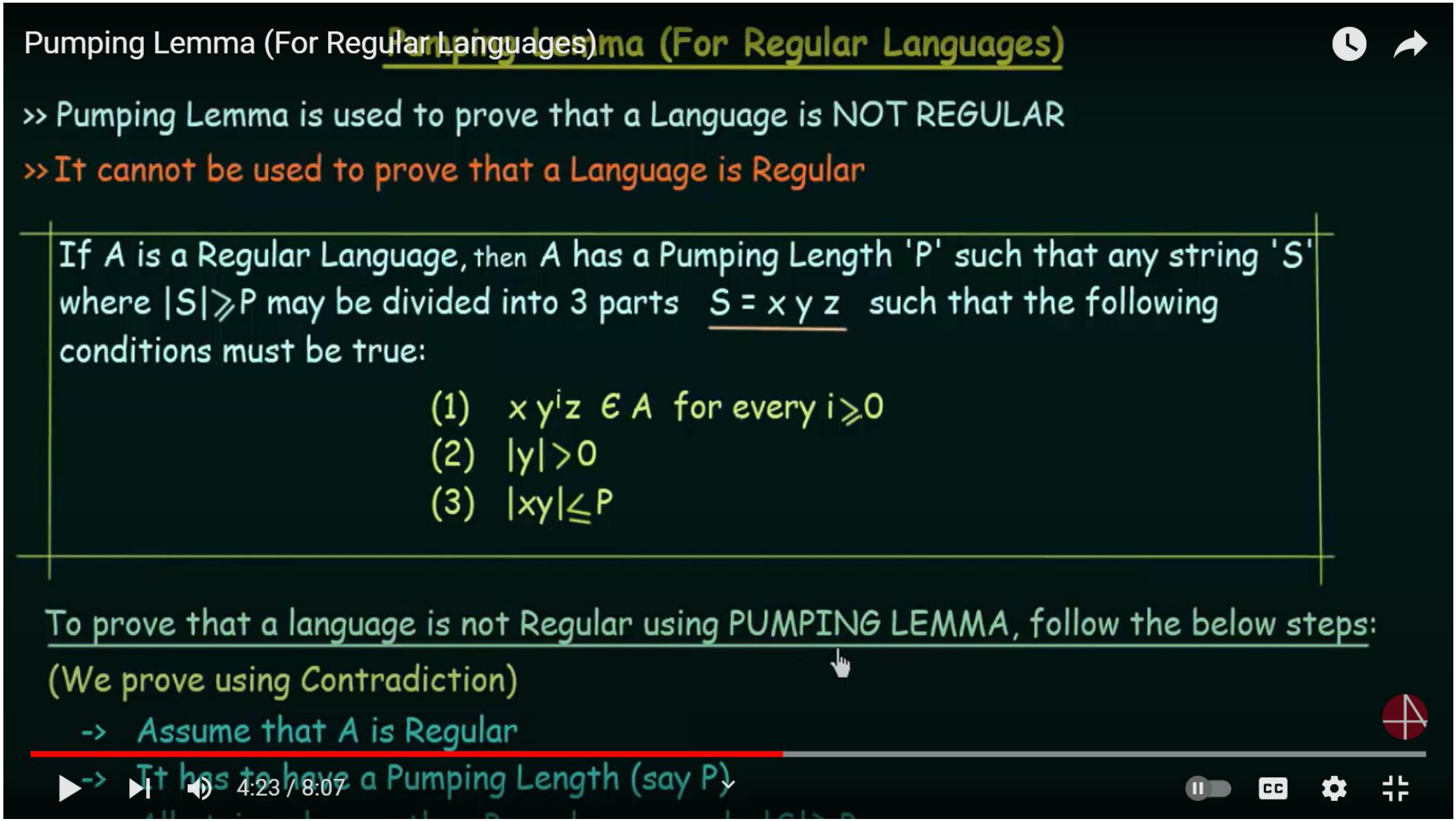

Lec-31: Pumping lemma for regular languages in TOC with examples
integer 'n' (Pumping length) such that any string W ∈ L
has length greater than equal to 'n' i.e. |W| ≥ n then w can
be divided into three parts, w= xyz satisfy following conditions
i) for each i ≥0, xy^i z ∈ L ✓
ii) |y| >0 and (bb)^{i+1}
iii) |xy| ≤n
L → PLT Pass Undecidable Reg./ Not Reg.
fail
Non Reg. Decidable.
✓ a^n b^{2n} n≥0
aa bbbb ∈L
aa bbbb bb ∉L
a ab ab bbb ∉L
11:37 / 11:59


Pumping Lemma (For Regular Languages)
Pumping Lemma (For Regular Languages)
>> Pumping Lemma is used to prove that a Language is NOT REGULAR
>> It cannot be used to prove that a Language is Regular
If A is a Regular Language, then A has a Pumping Length 'P' such that any string 'S' where |S| ≥ P may be divided into 3 parts S = x y z such that the following conditions must be true:
(1) x y^i z ∈ A for every i ≥ 0
(2) |y| > 0
(3) |xy| ≤ P
To prove that a language is not Regular using PUMPING LEMMA, follow the below steps:
(We prove using Contradiction)
-> Assume that A is Regular
-> It has to have a Pumping Length (say P)
4:23 / 8:07
CC

Pumping Lemma (For Context Free Languages)

Pumping Lemma (for CFL) is used to prove that a language is NOT Context Free

If A is a Context Free Language, then, A has a Pumping Length 'P' such that any string 'S', where $|S| \geqslant P$ may be divided into 5 pieces $S = uvxyz$ such that the following conditions must be true:

(1) $u\,v^i x\,y^i z$ is in A for every $i \geqslant 0$

(2) $|v\,y| > 0$

(3) $|v\,x\,y| \leqslant P$

Closure Properties of Regular Languages

1) Union $(L_1 \cup L_2)$
2) Concatenation $(L_1 \cdot L_2)$
3) Closure $(*)$ L^*
4) Complementation $\overline{L} = \Sigma^* - L$
5) Intersection $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
6) Difference $L_1 - L_2 = L_1 \cap \overline{L_2}$

Reversal $(L)^R$

Homomorphism

Reverse Homomorphism

Quotient operation

INIT

Substitution

Finite Union *

$L_1 = a$

$L = DFA$

Integer $\mathbb{Z}$ $-\infty$ -5 -4 .. 0 1 2

$1 + 2 = 3$

$\frac{3}{2} = 1.5$

RE . RE

RL

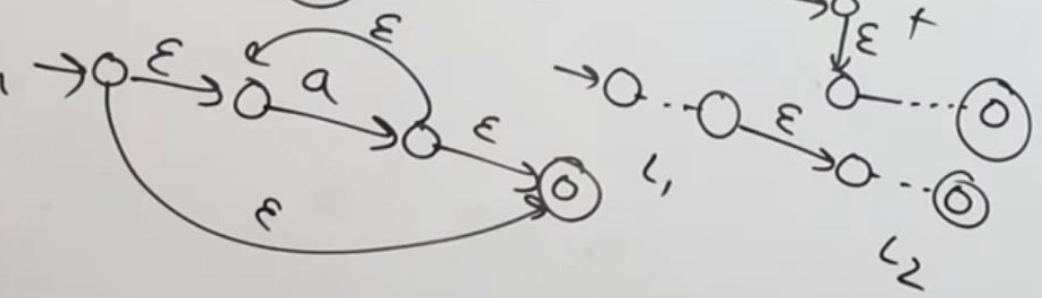
L_1 L

L_2

$L_1 \rightarrow RE$ + $L_2 \rightarrow RE$

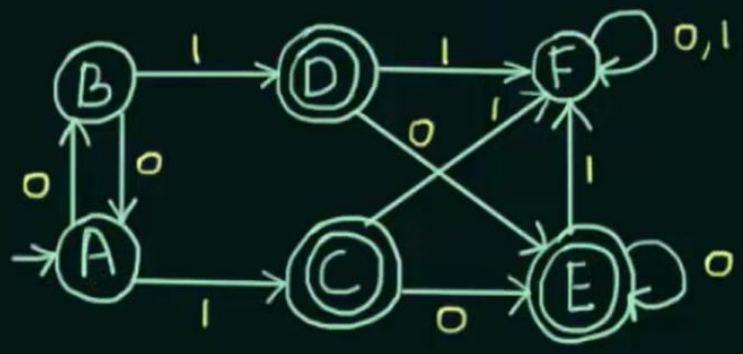
L_1^*

L_1 L_2 $0 = 0/P$

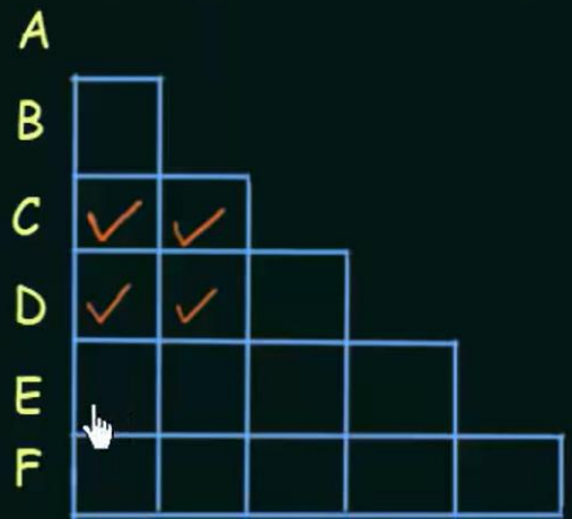


Minimization of DFA - Table Filling Method

(Myhill-Nerode Theorem)



A B C D E F

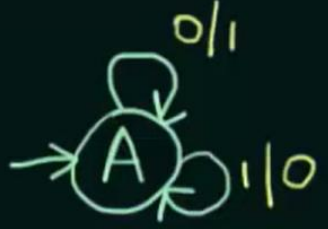


Steps:

1) Draw a table for all pairs of states (P,Q)
2) Mark all pairs where $P \in F$ and $Q \notin F$
3) If there are any Unmarked pairs (P,Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked, then mark [P,Q] where 'x' is an input symbol
   REPEAT THIS UNTIL NO MORE MARKINGS CAN BE MADE
4) Combine all the Unmarked Pairs and make them a single state in the minimized DFA

Construction of Mealy Machine

Ex-1) Construct a Mealy Machine that produces the 1's Complement of any binary input string.

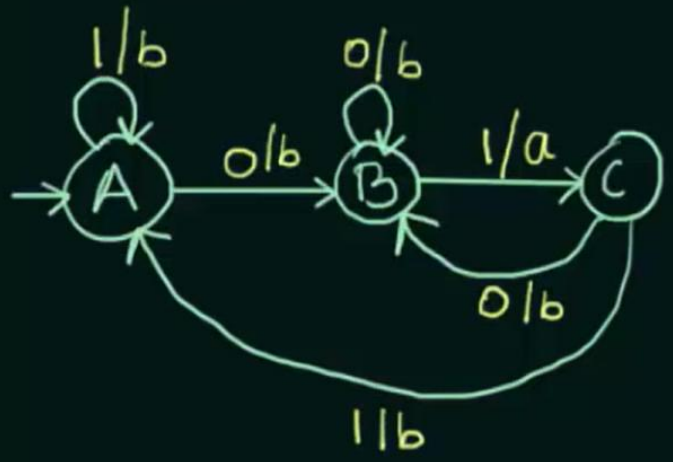


10100

01011

Ex-2) Construct a Mealy Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string.

$\Sigma = \{0,1\}$ $\Delta = \{a,b\}$



0110

baba

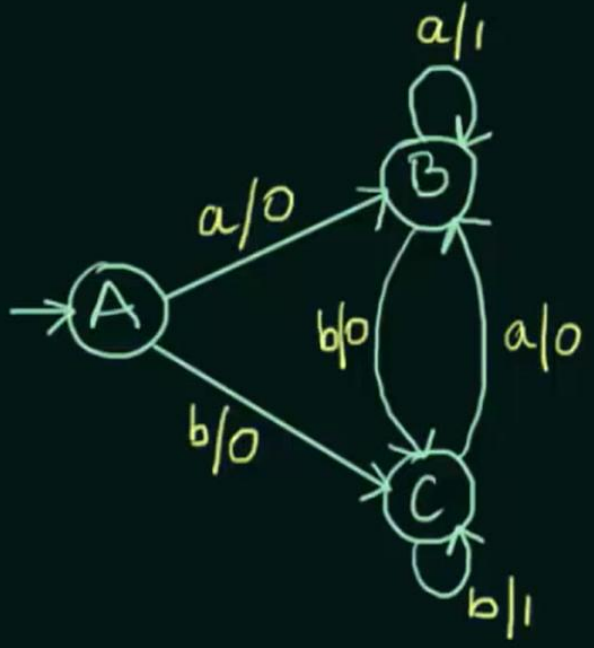
1000

bbbb

Construction of Mealy Machine - Examples (Part-1)

Design a Mealy Machine accepting the language consisting of strings from Σ^*, where Σ = {a,b} and the strings should end with either aa or bb

aa - 1
bb - 1



abb

baa

ba
00

aa
01

Construction of Mealy Machine -Examples (Part-2)

Construct a Mealy Machine that gives 2's Complement of any binary input. (Assume that the last carry bit is neglected)

2's complement = 1's complement + 1

MSB ← LSB

Eg. 10100

1's. 01011

+1

2's = 01100

Eg. 11100

1's. 00011

+ 1

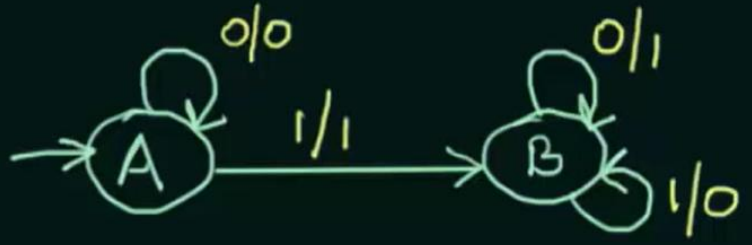
2's: 00100

Eg. 1110(1

1's. 0000

2's +1

0001

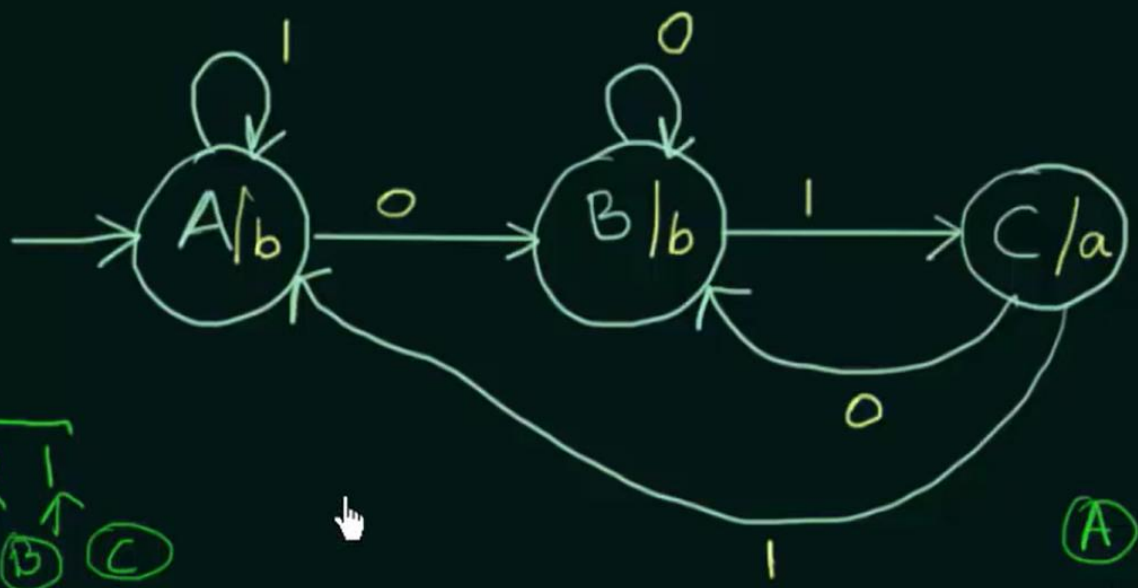


Construction of Moore Machine

Construct a Moore Machine that prints 'a' whenever the sequence '01' is encountered in any input binary string

$\Sigma = \{0, 1\}$

$\Delta = \{a, b\}$



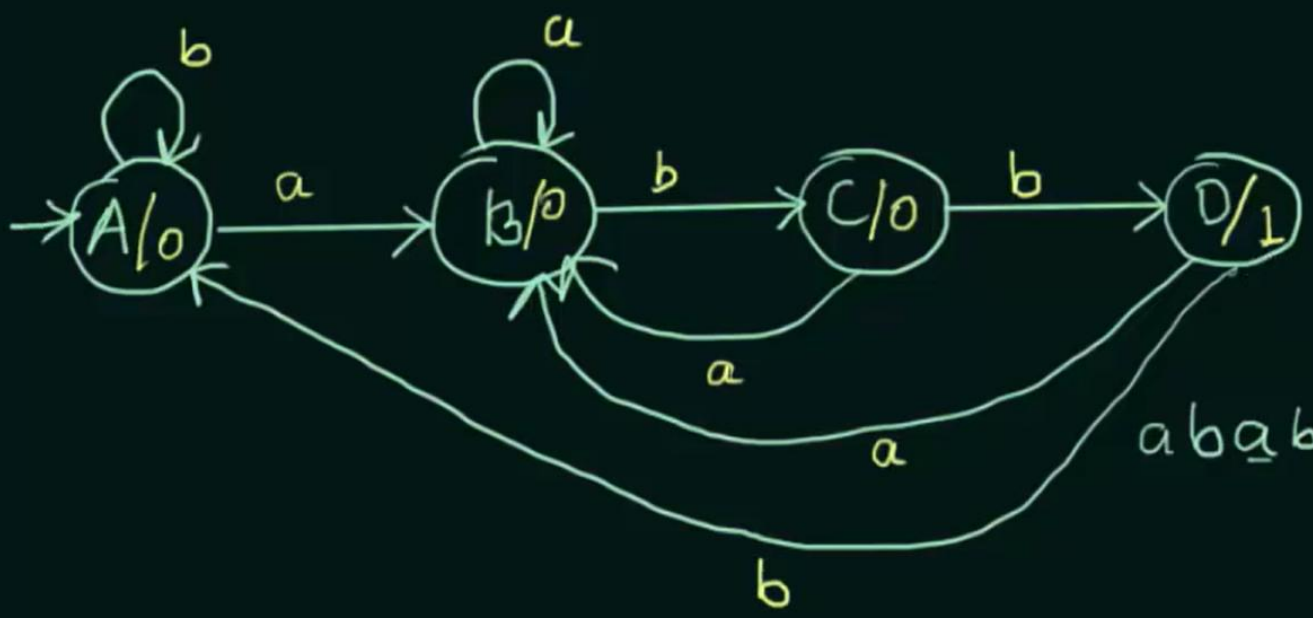
0101

01101

Construction of Moore Machine - Examples (Part-1)

Construct a Moore Machine that counts the occurences of the sequence 'abb' in any input strings over {a,b}

$\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$



abaabb

abbabb

abbbabb

For the following Moore Machine the input alphabet is $\Sigma = \{a,b\}$ and the output alphabet is $\Delta = \{0,1\}$. Run the following input sequences and find the repective outputs:

(i) aabab (ii) abbb (iii) ababb

States	a	b	Outputs
$\rightarrow q_0$	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

i) a a b a b

q_0 q_1 q_2 q_4 q_0 q_2

0 0 1 0 0 1

ii) a b b b

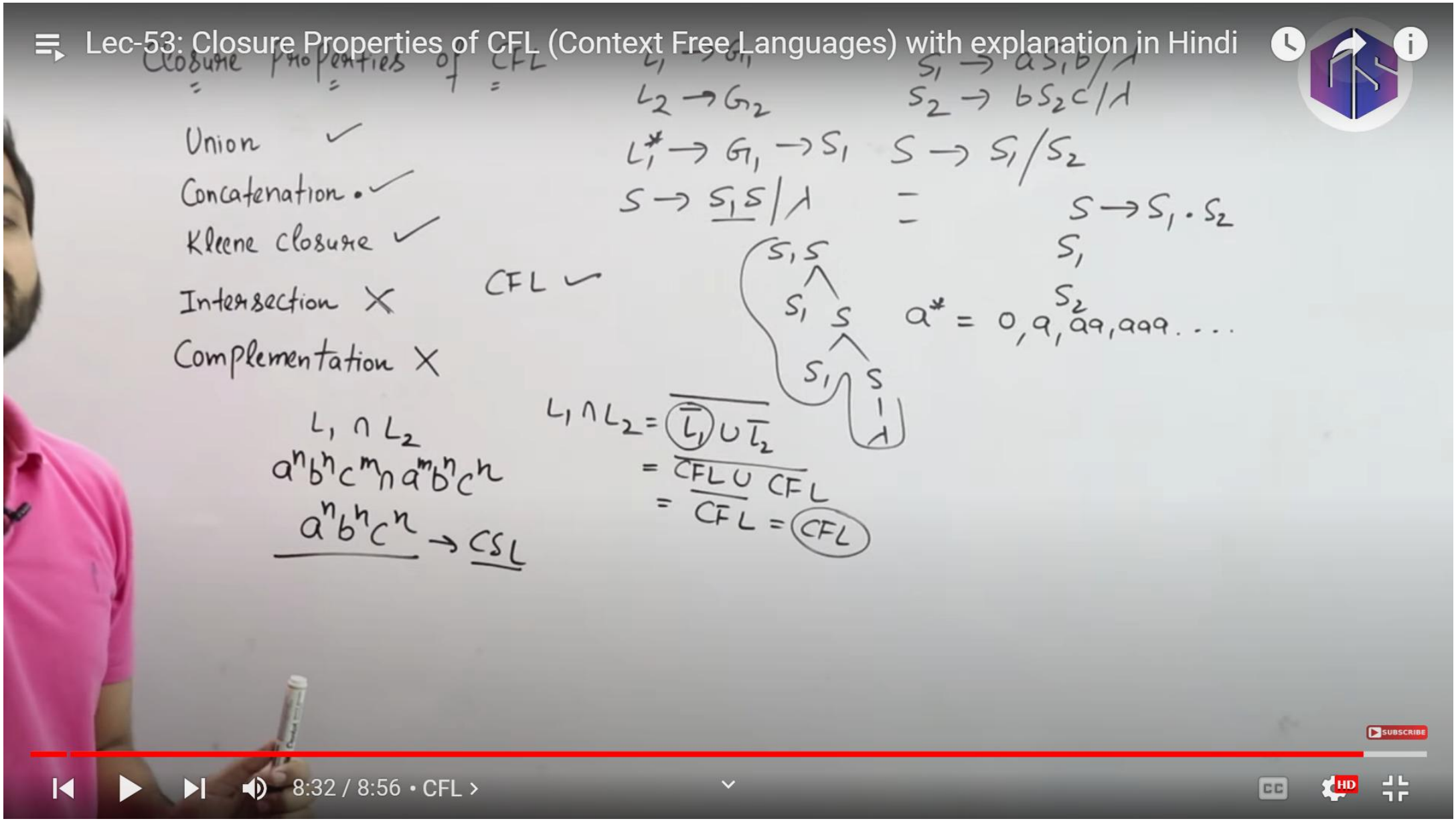
q_0 q_1 q_3 q_4 q_0

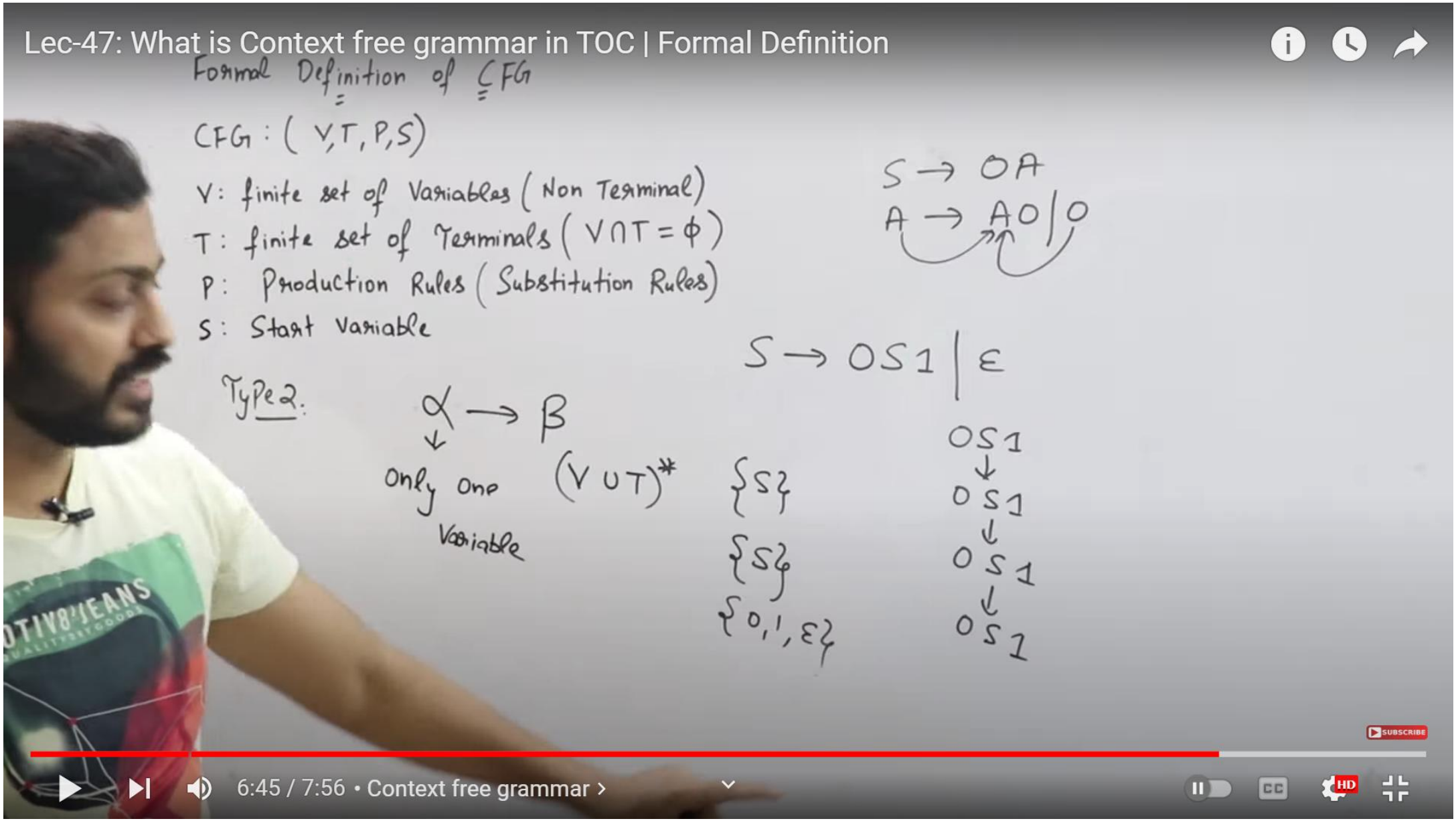
0 0 0 0 0

iii) a b a b b

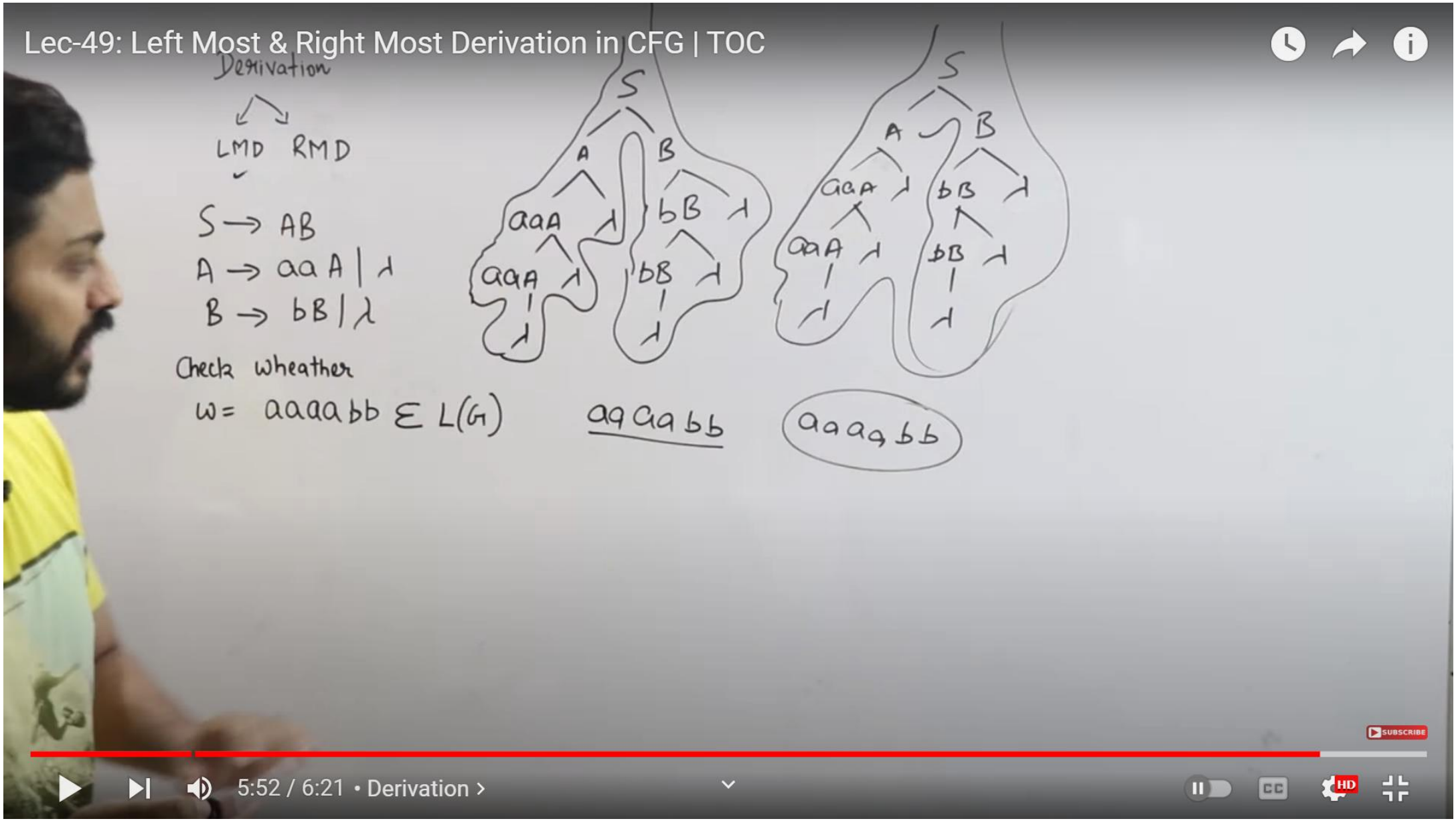
q_0 q_1 q_3 q_4 q_0 q_2

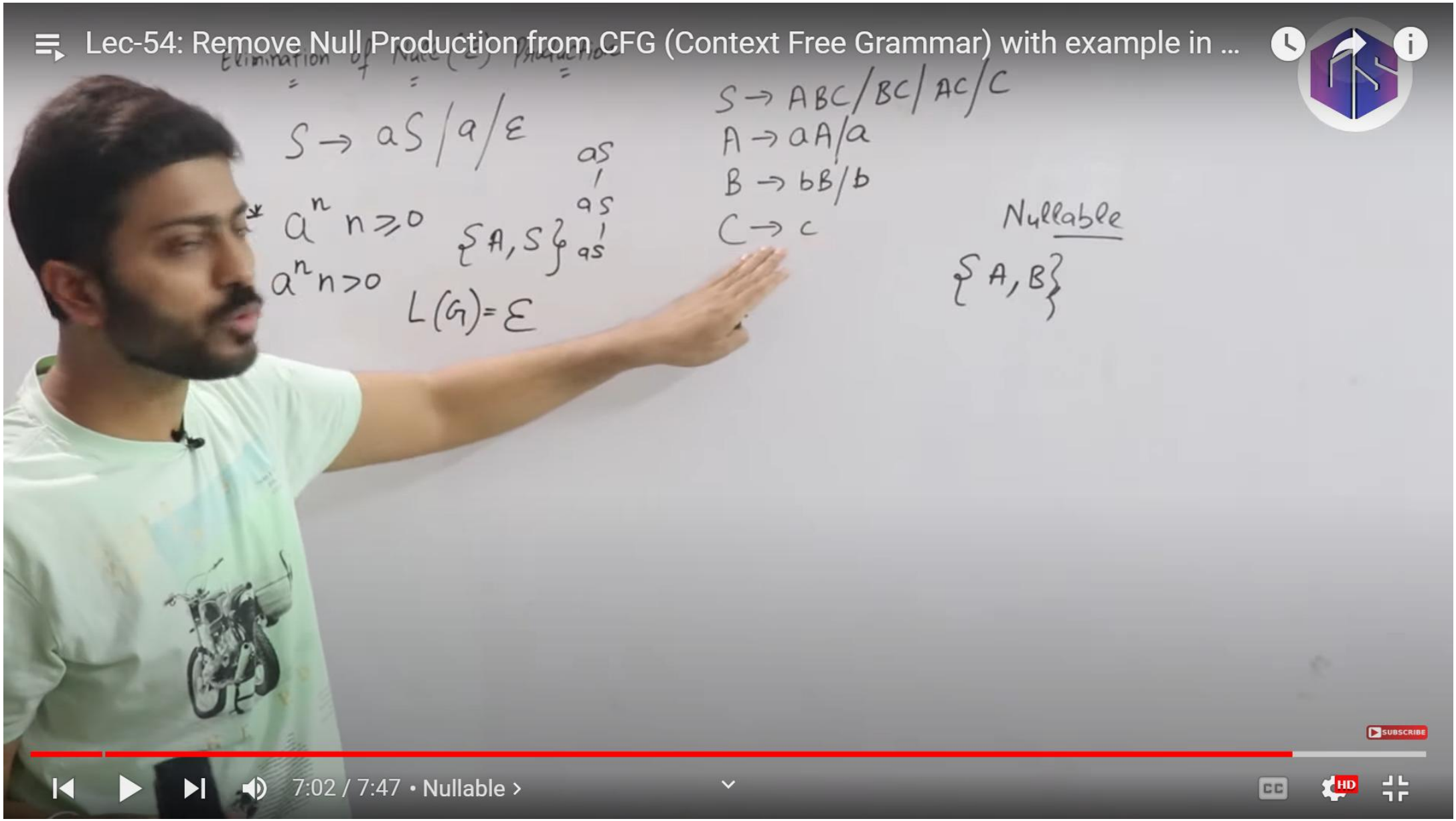
0 0 0 0 0 1

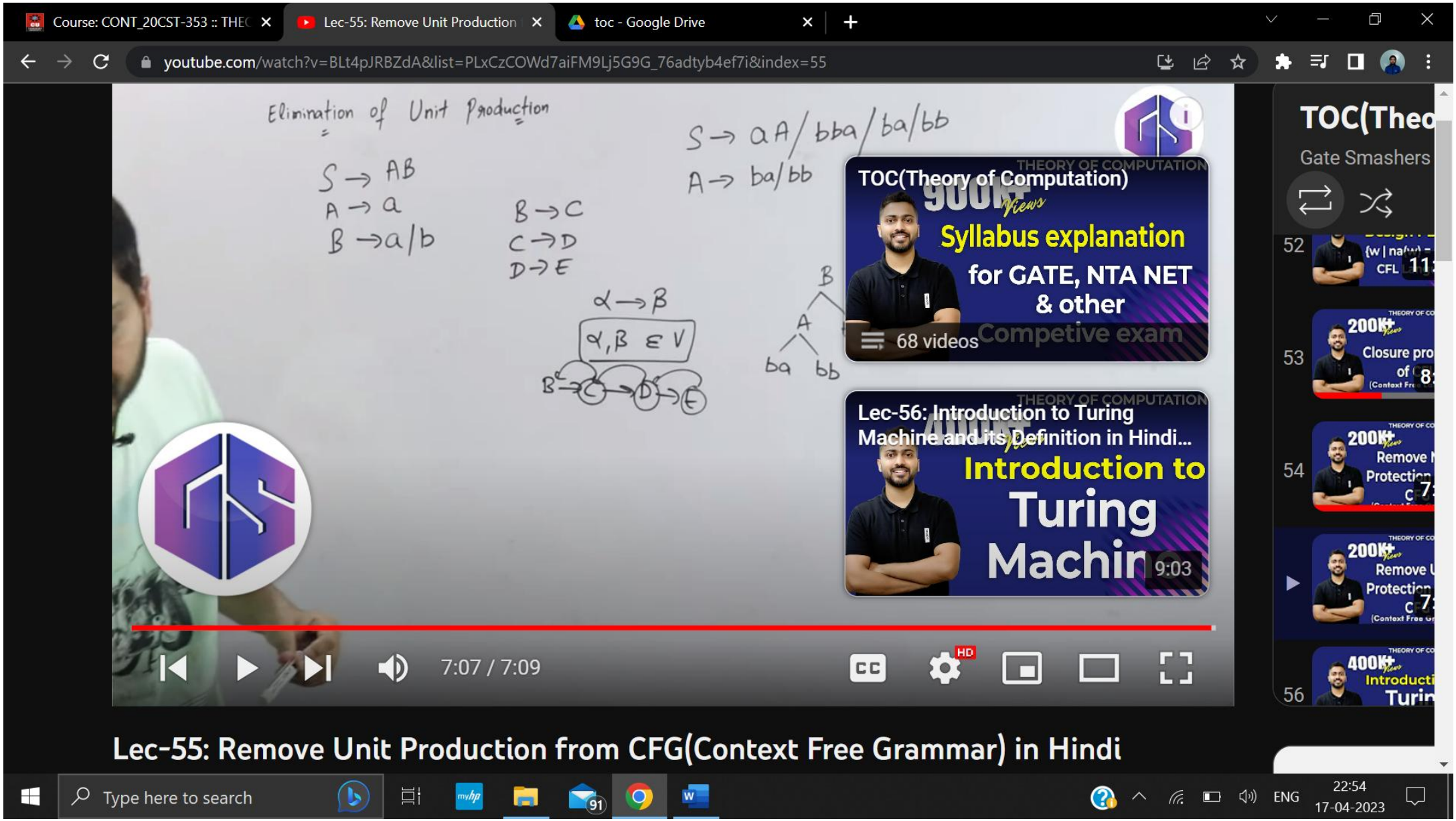

Lec-53: Closure Properties of CFL (Context Free Languages) with explanation in Hindi
Closure Properties of CFL
Union ✓
Concatenation . ✓
Kleene closure ✓
Intersection ✗
Complementation ✗
CFL ✓
L1 ∩ L2
anbncmn ambncn
anbncn → CSL
L1 → G1
L2 → G2
L1* → G1 → S1
S → S1S | λ
S1 → aS1b | λ
S2 → bS2c | λ
S → S1 / S2
= S → S1 . S2
S1
S2
a* = 0, a, aa, aaa ....
L1 ∩ L2 = (L1' ∪ L2')'
= (CFL ∪ CFL)'
= CFL' = CFL
8:32 / 8:56 • CFL ›


Lec-47: What is Context free grammar in TOC | Formal Definition
Formal Definition of CFG
CFG : (V, T, P, S)
V : finite set of Variables (Non Terminal)
T : finite set of Terminals (V ∩ T = ϕ)
P : Production Rules (Substitution Rules)
S : Start Variable
S → OA
A → AO | O
S → OS1 | ε
Type 2.
α → β
only one Variable
(V ∪ T)*
{S}
{S}
{0, 1, ε}
OS1
OS1
OS1
OS1
6:45 / 7:56 • Context free grammar ›

Lec-49: Left Most & Right Most Derivation in CFG | TOC




Lec-54: Remove Null Production from CFG (Context Free Grammar) with example in ...
S → aS/a/ε
aS
a S
a S
a^n n≥0
{A, S}
a^n n>0
L(G)=ε
S → ABC/BC/AC/C
A → aA/a
B → bB/b
C → c
Nullable
{A, B}
SUBSCRIBE
7:02 / 7:47 • Nullable ›
CC
HD


Course: CONT_20CST-353 :: THEC
Lec-55: Remove Unit Production
toc - Google Drive
youtube.com/watch?v=BLt4pJRBZdA&list=PLxCzCOWd7aiFM9Lj5G9G_76adtyb4ef7i&index=55
Elimination of Unit Production
S → AB
A → a
B → a/b
B → C
C → D
D → E
S → aA/bba/ba/bb
A → ba/bb
α → β
α,β ∈ V
TOC(Theory of Computation)
900K+ Views
Syllabus explanation
for GATE, NTA NET & other
Competive exam
68 videos
Lec-56: Introduction to Turing Machine and its Definition in Hindi...
400K+ Views
Introduction to
Turing
Machin
9:03
7:07 / 7:09
TOC(Theo
Gate Smashers
52
53
54
56
Lec-55: Remove Unit Production from CFG(Context Free Grammar) in Hindi
Type here to search
ENG
22:54
17-04-2023

Lec-64: CNF Vs GNF | Chomsky vs Greibach Normal Form | CFG in TOC

CNF
(Chomsky Normal Form)

$S \to AB$
$A \to BB/a$
$B \to AB/b$

1) $A \to BC$ OR $A \to a$ $\{A, B, C \in \text{Variables}\}$ $\{a \in \text{Terminal}\}$

2) No. of steps required to generate a string of length 'n' is "$2|n| - 1$"
$2 \times 2 - 1 = 3$ ab

3) Used in Membership Algo. CYK.

S
A → AB
A → a
B → b

4) Derivation or Parse tree obtained from CNF is always Binary Tree.

A
↓
AB
↓
aB
↓
ab

5) Length of each Production Restricted

GNF
(Greibach Normal Form)

1) $S \to a\,ABCDEF$ $A \to a\,x$
$A \to a$ where $\{x \in V^*\}$ $\{a \in T\}$

2) No. of steps required to generate a string of length 'n' is 'n'

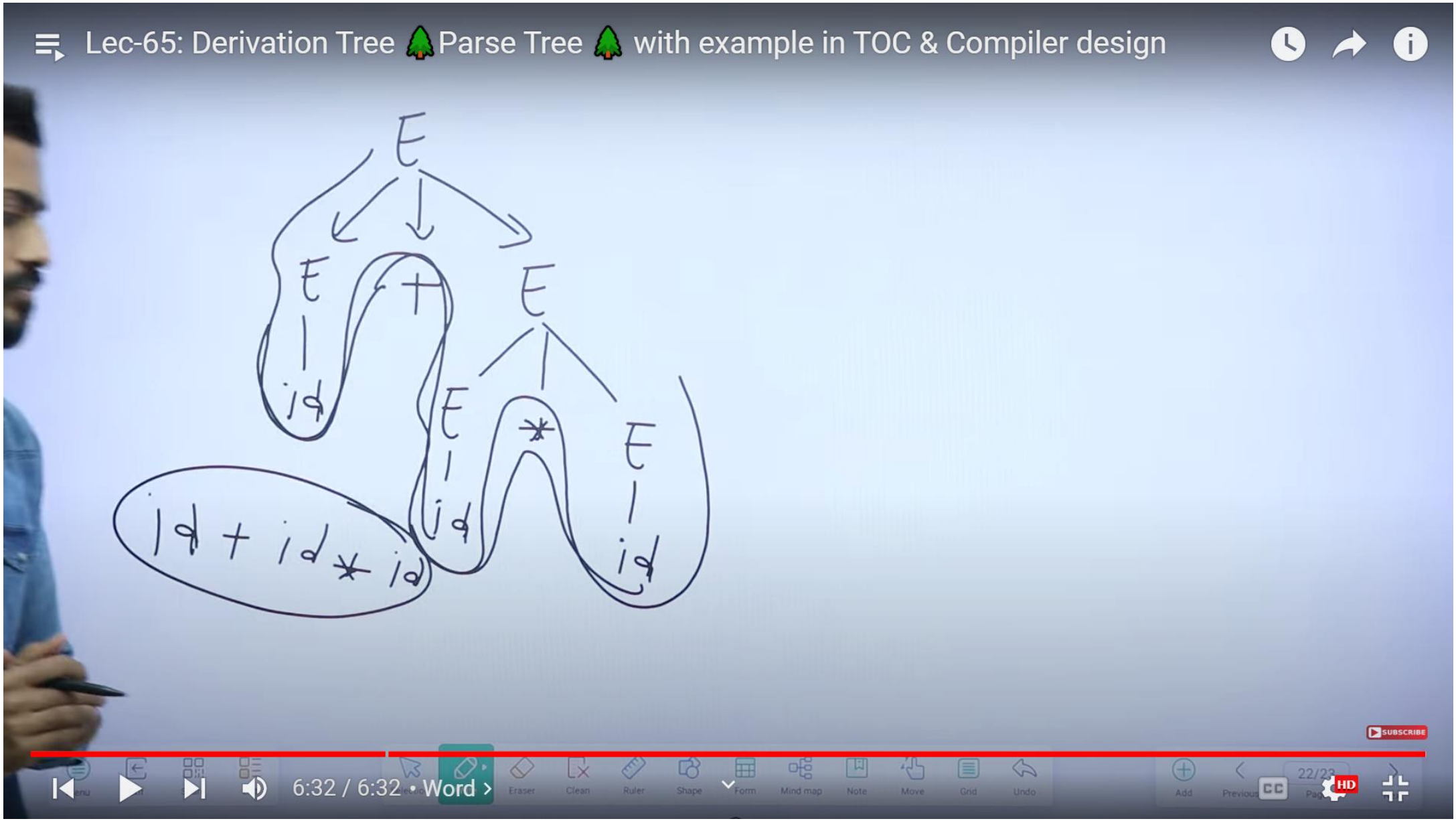
3) Used to Convert CFG to PDA

4) Not always

A → aB
B → b

A
↓
aB
↓
b

5) Not Restricted


Lec-65: Derivation Tree 🌲Parse Tree 🌲 with example in TOC & Compiler design
E
E
+
E
id
E
*
E
id
id
id + id * id
SUBSCRIBE
6:32 / 6:32 • Word ›
Eraser
Clean
Ruler
Shape
Form
Mind map
Note
Move
Grid
Undo
Add
Previous
22/22
CC
HD

Lec-66: Recursive vs Non-Recursive CFG with examples | Classification of CFG

- Recursive CFG generates Infinite number of strings.

for example:

1) S->Sa

S->b

{ b, ba, bba, bbba .... }

2) S->aS

S->b

{ b, ab, aab, aaab ... }

3) S-> Aa

A->Ab|c

{ ca, cba, ccba, ccbba ... }

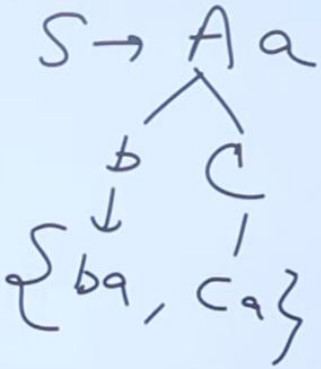
- Non-Recursive CFG generates finite number of strings.

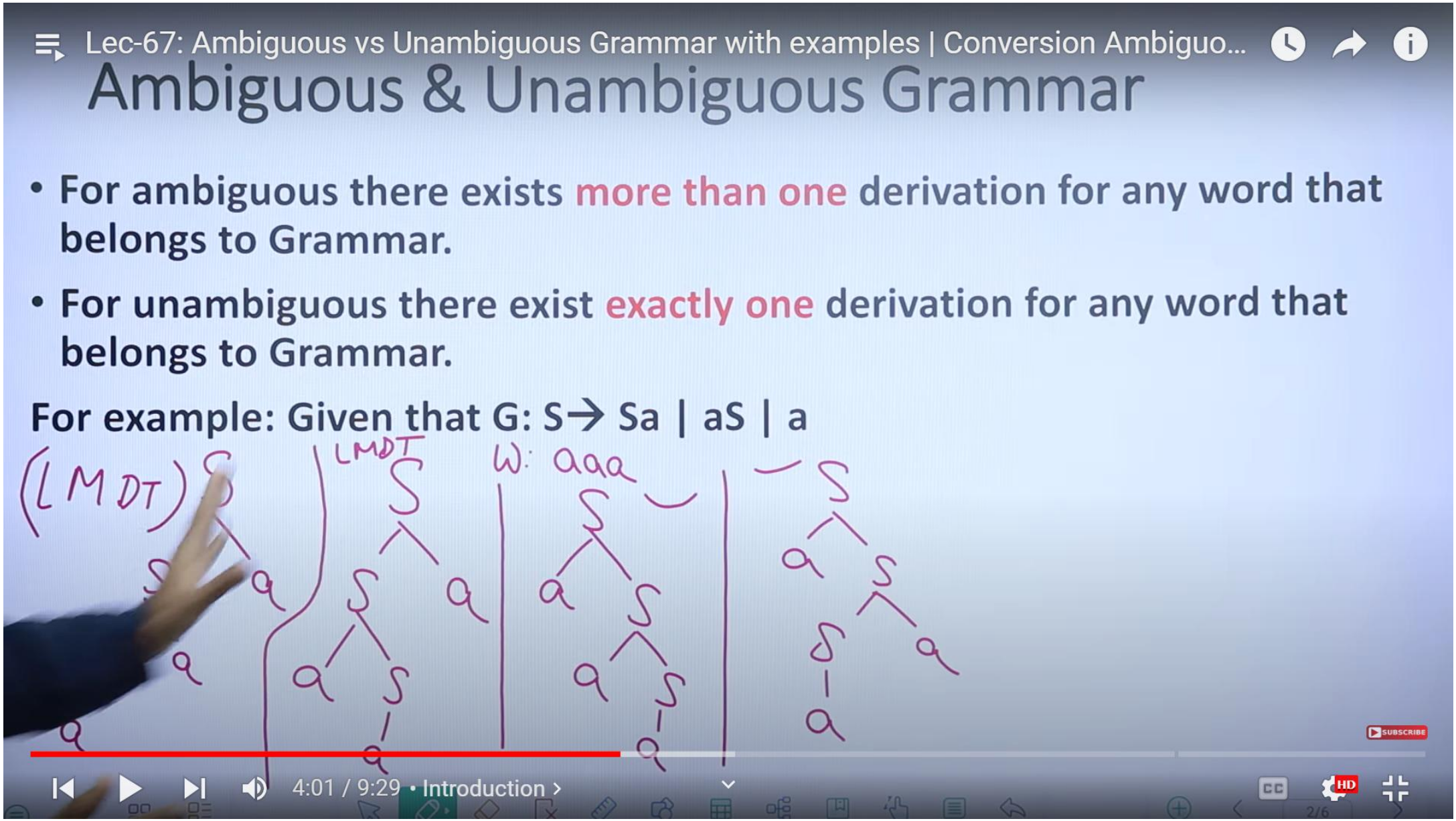
For example:

S->Aa

A->b|c

{ ba, ca }




Lec-67: Ambiguous vs Unambiguous Grammar with examples | Conversion Ambiguo...
Ambiguous & Unambiguous Grammar
• For ambiguous there exists more than one derivation for any word that belongs to Grammar.
• For unambiguous there exist exactly one derivation for any word that belongs to Grammar.
For example: Given that G: S→ Sa | aS | a
(LMDT)
LMDT
W: aaa
4:01 / 9:29 • Introduction ›
SUBSCRIBE
HD
CC

Lec-67: Ambiguous vs Unambiguous Grammar with examples | Conversion Ambiguo...

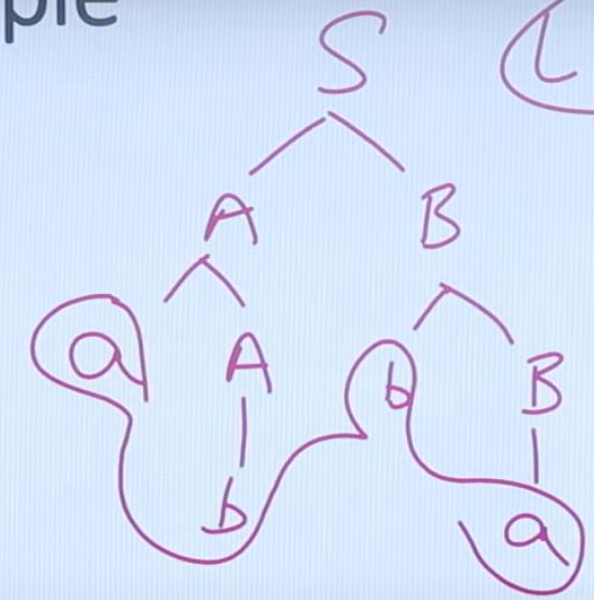
More Example

G_1: S → AB

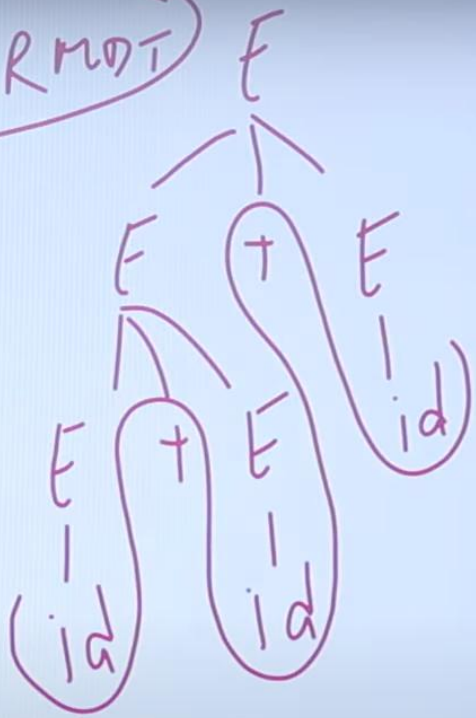
A → aA | b

B → bB | a

W: abba

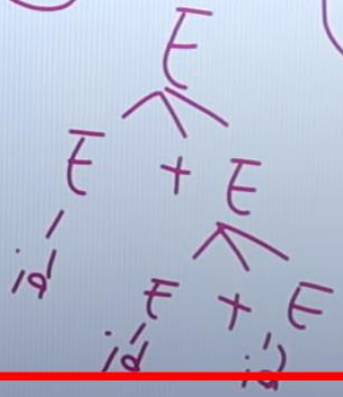


LMDT = RMDT



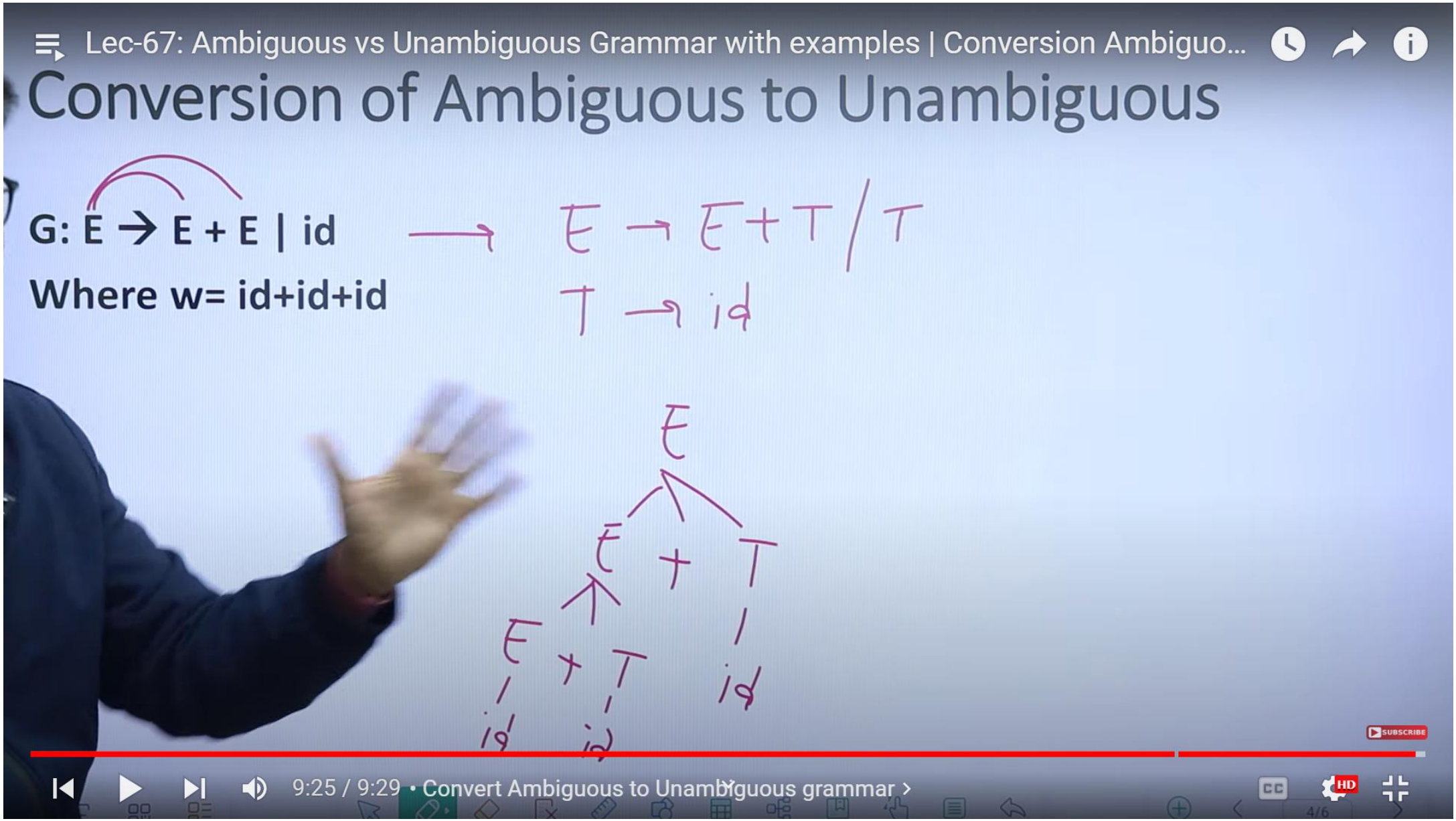
G_2: E → E + E | id

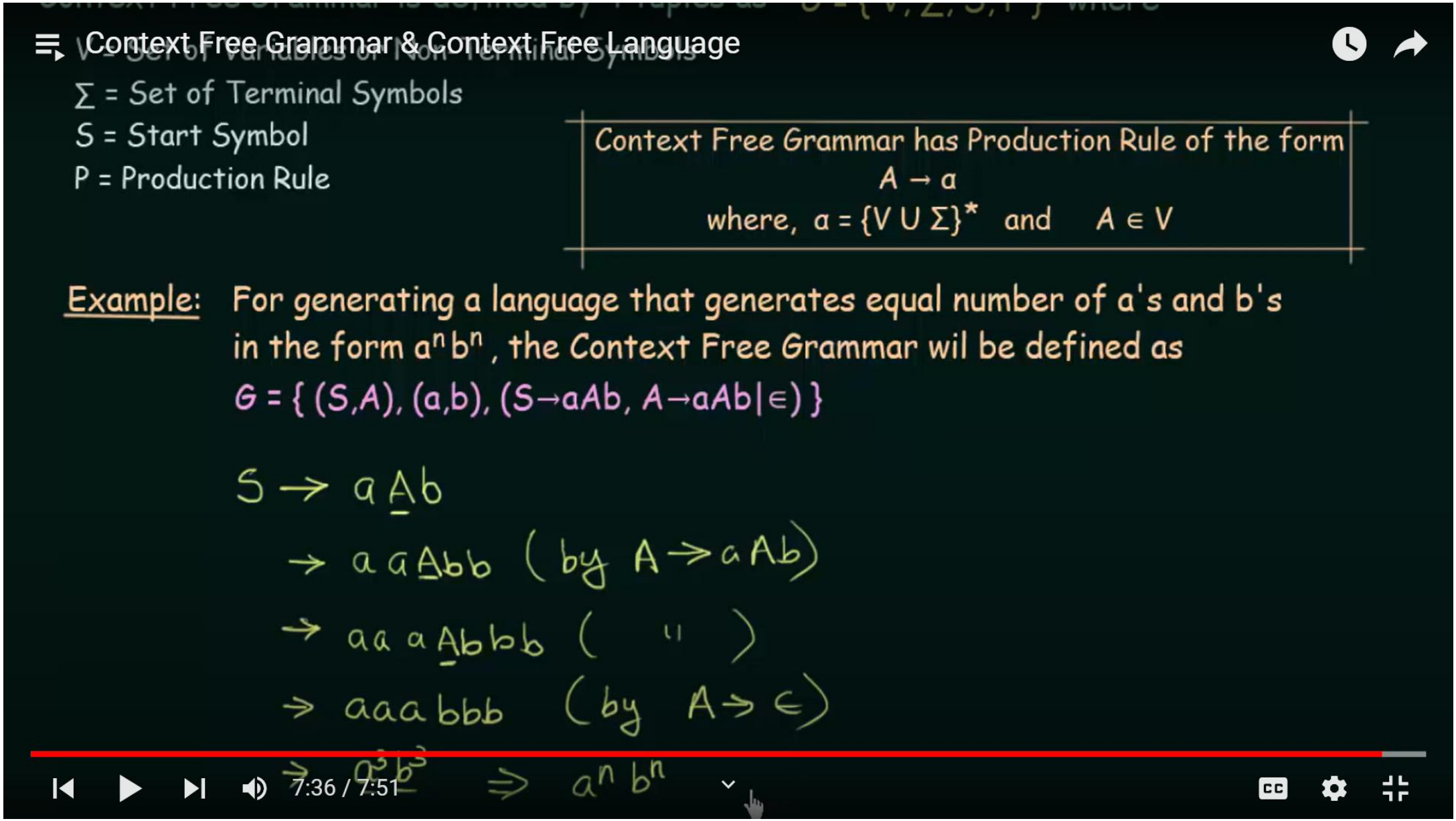
W: id+id+id

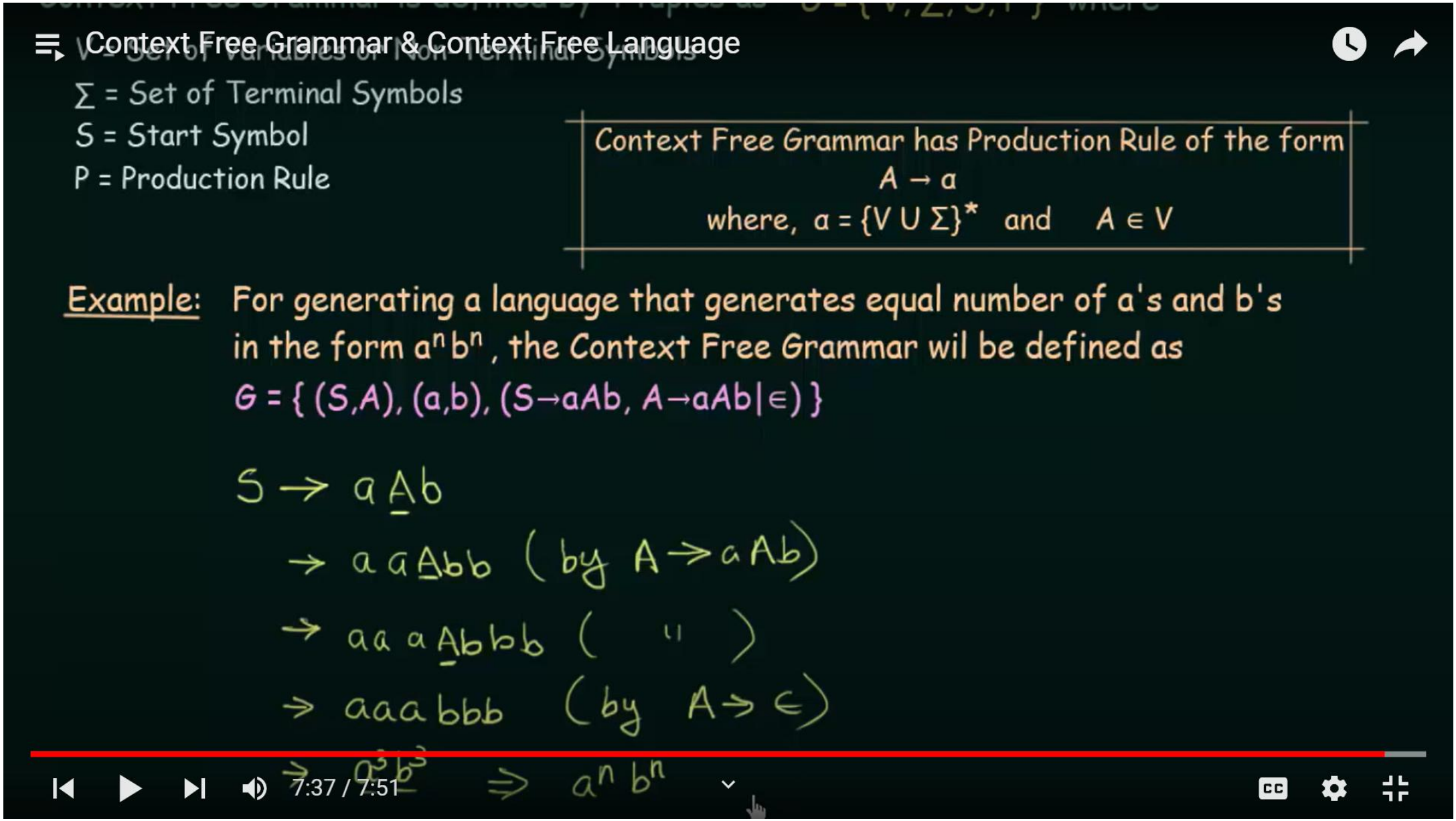


SUBSCRIBE

7:44 / 9:29 • Introduction >


Lec-67: Ambiguous vs Unambiguous Grammar with examples | Conversion Ambiguo...
Conversion of Ambiguous to Unambiguous
G: E → E + E | id
Where w= id+id+id
E → E+T / T
T → id
E
E + T
E + T
id
id
id
SUBSCRIBE
9:25 / 9:29 • Convert Ambiguous to Unambiguous grammar ›
CC
HD
4/6

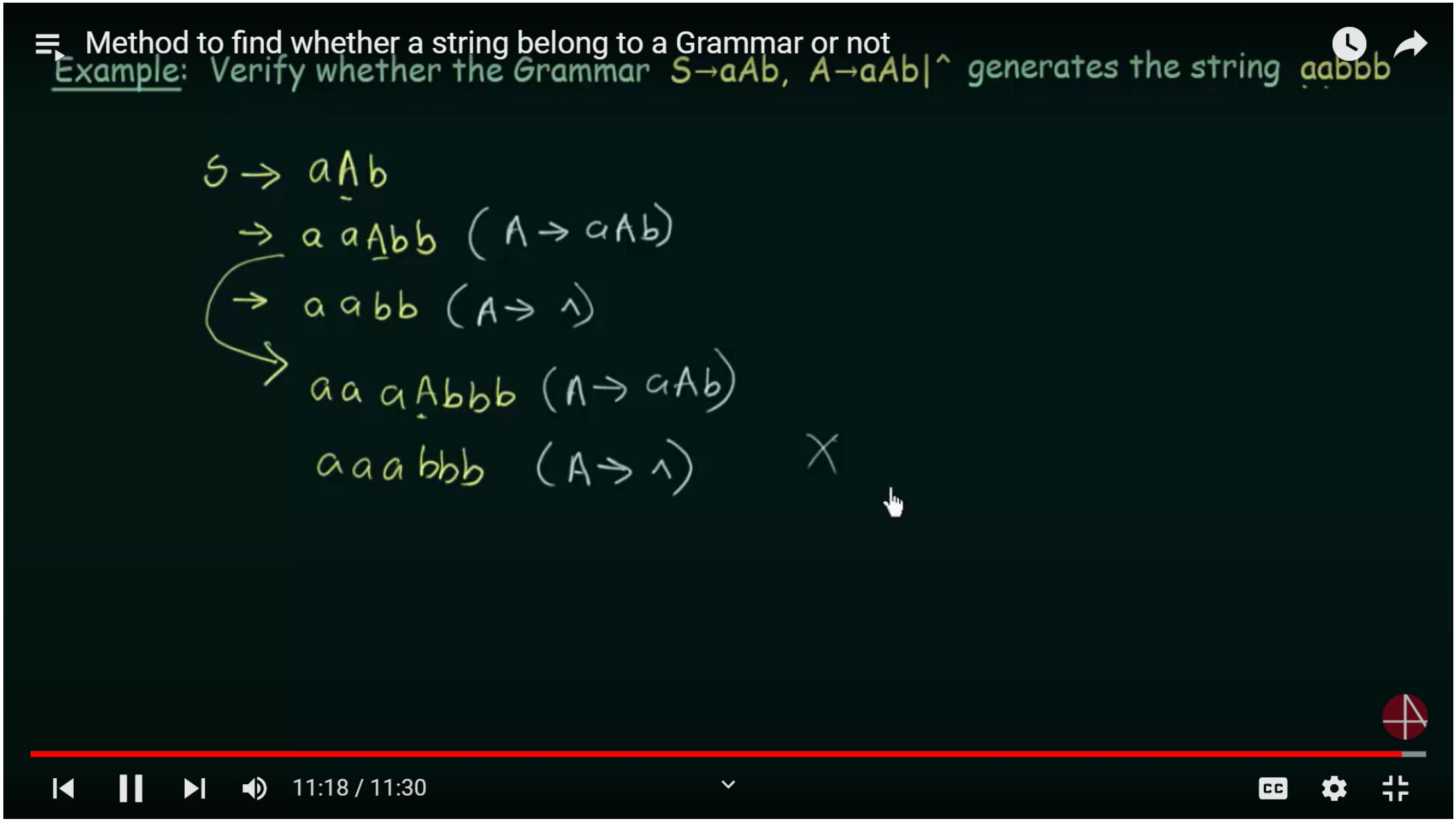

Context Free Grammar & Context Free Language
V = Set of Variables or Non-Terminal Symbols
Σ = Set of Terminal Symbols
S = Start Symbol
P = Production Rule
Context Free Grammar has Production Rule of the form
A → α
where, α = {V U Σ}* and A ∈ V
Example: For generating a language that generates equal number of a's and b's in the form aⁿbⁿ, the Context Free Grammar wil be defined as
G = { (S,A), (a,b), (S→aAb, A→aAb|∈) }
S → aAb
→ aaAbb (by A→aAb)
→ aaaAbbb (")
→ aaabbb (by A→∈)
⇒ a³b³ ⇒ aⁿbⁿ
7:36 / 7:51

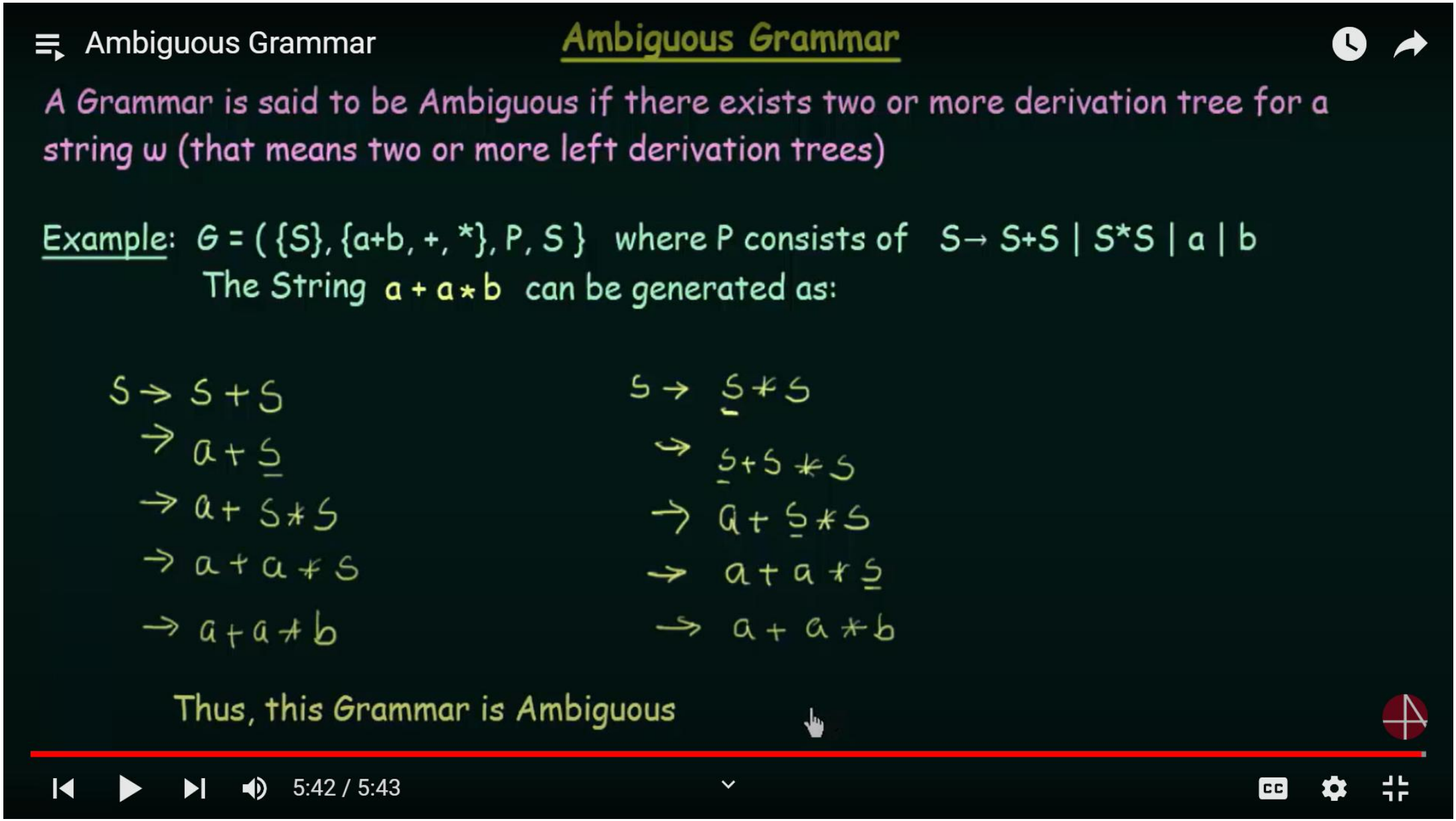

Context Free Grammar & Context Free Language
V = Set of Variables or Non-Terminal Symbols
Σ = Set of Terminal Symbols
S = Start Symbol
P = Production Rule
Context Free Grammar has Production Rule of the form
A → α
where, α = {V U Σ}* and A ∈ V
Example: For generating a language that generates equal number of a's and b's in the form aⁿbⁿ, the Context Free Grammar wil be defined as
G = { (S,A), (a,b), (S→aAb, A→aAb|∈) }
S → aAb
→ aaAbb (by A→aAb)
→ aaaAbbb (")
→ aaabbb (by A→∈)
⇒ a³b³ ⇒ aⁿbⁿ
7:37 / 7:51

1) Start with the Start Symbol and choose the closest production that matches to the given string.
2) Replace the Variables with its most appropriate production. Repeat the process until the string is generated or until no other productions are left.

Example: Verify whether the Grammar $S \rightarrow 0B|1A$, $A \rightarrow 0|0S|1AA|\hat{}$, $B \rightarrow 1|1S|0BB$ generates the string 00110101

$S \rightarrow 0B$ ($S \rightarrow 0B$)
$\rightarrow 00BB$ ($B \rightarrow 0BB$)
$\rightarrow 001B$ ($B \rightarrow 1$)
$\rightarrow 0011S$ ($B \rightarrow 1S$)
$\rightarrow 00110B$ ($S \rightarrow 0B$)
$\rightarrow 001101S$ ($B \rightarrow 1S$)
$\rightarrow 0011010B$ ($S \rightarrow 0B$)
$\rightarrow 00110101$ ($B \rightarrow 1$)


Method to find whether a string belong to a Grammar or not
Example: Verify whether the Grammar S→aAb, A→aAb|^ generates the string aabbb
S → aAb
→ a aAbb (A → aAb)
→ a abb (A → ^)
aa aAbbb (A → aAb)
aaa bbb (A → ^)
11:18 / 11:30


Ambiguous Grammar
Ambiguous Grammar
A Grammar is said to be Ambiguous if there exists two or more derivation tree for a string w (that means two or more left derivation trees)
Example: G = ({S}, {a+b, +, *}, P, S } where P consists of S→ S+S | S*S | a | b
The String a + a * b can be generated as:
S → S + S
→ a + S
→ a + S * S
→ a + a * S
→ a + a * b
S → S * S
→ S + S * S
→ a + S * S
→ a + a * S
→ a + a * b
Thus, this Grammar is Ambiguous
5:42 / 5:43

Example: Find a reduced grammar equivalent to the grammar G, having production rules

P: $S \to AC|B,\ A \to a,\ C \to c|BC,\ E \to aA|e$

Phase 1:

$T = \{a, c, e\}$

$W_1 = \{A, C, E\}$

$W_2 = \{A, C, E, S\}$

$W_3 = \{A, C, E, S\}$

$G' = \{(A, C, E, S), \{a, c, e\}, P, (S)\}$

P: $S \to AC,\ A \to a,\ C \to c,\ E \to aA|e$

Phase 2:

$Y_1 = \{S\}$

$Y_2 = \{S, A, C\}$

$W_2 = \{A, C, E, S\}$

$W_3 = \{A, C, E, S\}$

$G' = \{(A, C, E, S), \{a, c, e\}, P, (S)\}$

P: $S \to AC,\ A \to a,\ c \to c,\ E \to aA \mid e$

Phase 2 : $Y_1 = \{S\}$

$Y_2 = \{S, A, C\}$

$Y_3 = \{S, A, C, a, c\}$

$Y_4 = \{S, A, C, a, c\}$

$G'' = \{(A, C, S), \{a, c\}, P, \{S\}\}$

P: $S \to AC,\ A \to a,\ C \to c$

Simplification of Context Free Grammar

Removal of Unit Productions

Any Production Rule of the form $A \rightarrow B$ where $A, B \in$ Non Terminals is called Unit Production

Procedure for Removal

Step 1: To remove $A \rightarrow B$, add production $A \rightarrow x$ to the grammar rule whenever $B \rightarrow x$ occurs in the grammar. [$x \in$ Terminal, x can be Null]

Step 2: Delete $A \rightarrow B$ from the grammar.

Step 3: Repeat from Step 1 until all Unit Productions are removed.

Example: Remove Unit Productions from the Grammar whose production rule is given by

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow Z|b$, $Z \rightarrow M$, $M \rightarrow N$, $N \rightarrow a$

$Y \rightarrow Z$, $Z \rightarrow M$, $M \rightarrow N$

1) Since $N \rightarrow a$, we add $M \rightarrow a$

P: $S \rightarrow XY$, $X \rightarrow a$, $Y \rightarrow Z|b$, $Z \rightarrow M$, $M \rightarrow a$, $N \rightarrow a$

Example: Remove Unit Productions from the Grammar whose production rule is given by

P: $S \to XY$, $X \to a$, $Y \to Z|b$, $Z \to M$, $M \to N$, $N \to a$

$Y \to Z$, $Z \to M$, $M \to N$

1) Since $N \to a$, we add $M \to a$

P: $S \to XY$, $X \to a$, $Y \to Z|b$, $Z \to M$, $M \to a$, $N \to a$

2) Since $M \to a$, we add $Z \to a$

P: $S \to XY$, $X \to a$, $Y \to Z|b$, $Z \to a$, $M \to a$, $N \to a$

3) Since $Z \to a$, we add $Y \to a$

P: $S \to XY$, $X \to a$, $Y \to a|b$, $Z \to a$, $M \to a$, $N \to a$

Remove the Unreachable symbols

P: $S \to XY$, $X \to a$, $Y \to a|b$

Simplification of Context Free Grammar

Removal of Null Productions

In a CFG, a Non-Terminal Symbol 'A' is a nullable variable if there is a production $A \rightarrow \in$ or there is a derivation that starts at 'A' and leads to $\in$. (Like $A \rightarrow \ldots\ldots\ldots\ldots \rightarrow \in$)

Procedure for Removal:

Step 1: To remove $A \rightarrow \in$, look for all productions whose right side contains A

Step2: Replace each occurences of 'A' in each of these productions with $\in$

Step 3: Add the resultant productions to the Grammar

Example: Remove Null Productions from the following Grammar

$S \rightarrow ABAC,\quad A \rightarrow aA \mid \in,\quad B \rightarrow bB \mid \in,\quad C \rightarrow c$

$A \rightarrow \in \quad , \quad B \rightarrow \in$

1) To eliminate $A \rightarrow \in$

$S \rightarrow ABAC$

$S \to ABC \mid BAC \mid BC$

$A \to aA$

$A \to a$

New production: $S \to ABAC \mid ABC \mid BAC \mid BC$

$A \to aA \mid a$, $B \to bB \mid \epsilon$, $C \to c$

2) To eliminate $B \to \epsilon$

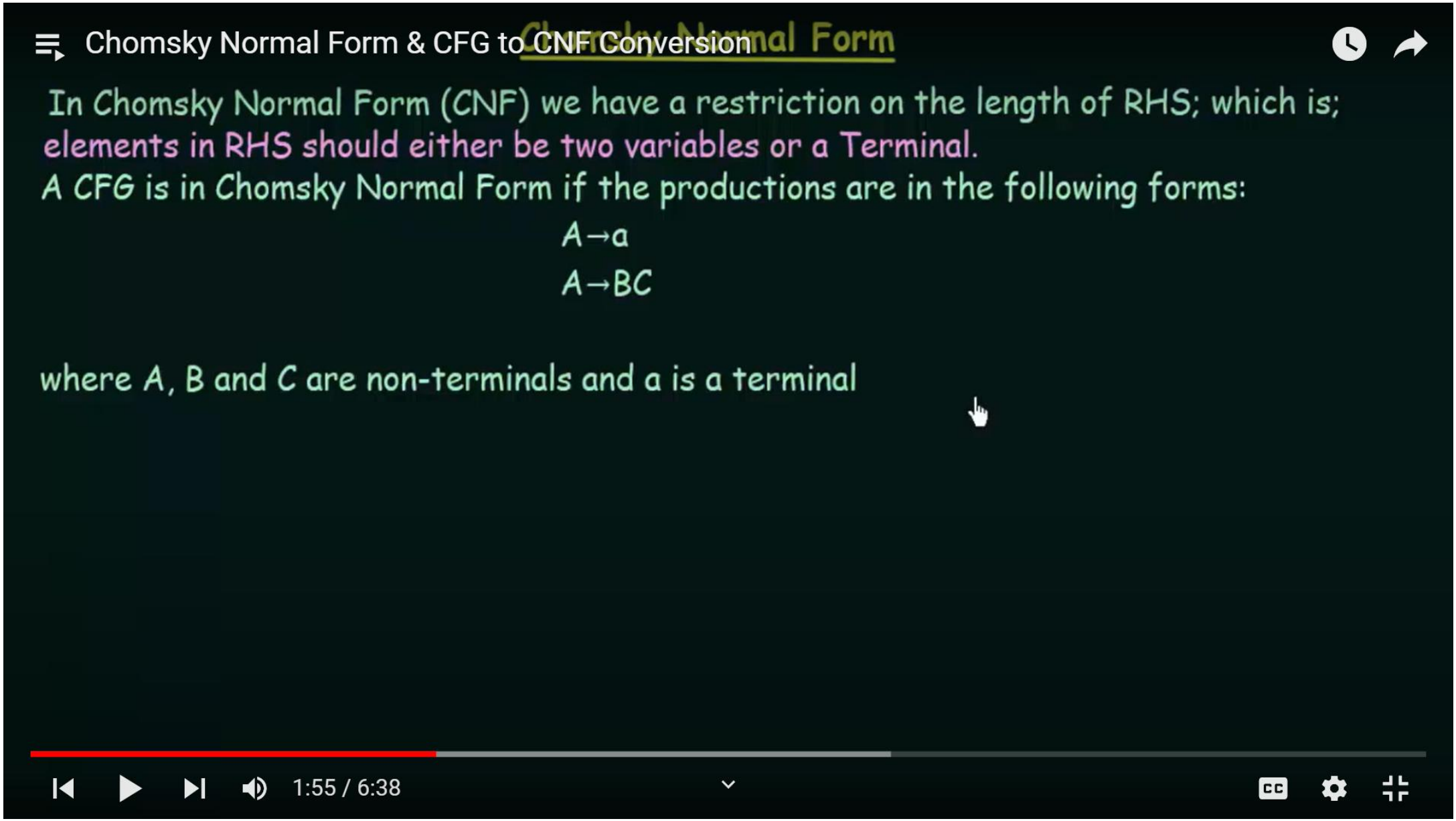
$S \to AAC \mid AC \mid C$, $B \to b$

New production: $S \to ABAC \mid ABC \mid BAC \mid BC \mid AAC \mid AC \mid C$

$A \to aA \mid a$

$B \to bB \mid b$

$C \to c$


Chomsky Normal Form & CFG to CNF Conversion
Chomsky Normal Form
In Chomsky Normal Form (CNF) we have a restriction on the length of RHS; which is;
elements in RHS should either be two variables or a Terminal.
A CFG is in Chomsky Normal Form if the productions are in the following forms:
A→a
A→BC
where A, B and C are non-terminals and a is a terminal
1:55 / 6:38

Steps to convert a given CFG to Chomsky Normal Form:

Step 1: If the Start Symbol S occurs on some right side, create a new Start Symbol S' and a new Production $S' \rightarrow S$.

Step 2: Remove Null Productions. (Using the Null Production Removal discussed in previous Lecture)

Step 3: Remove Unit Productions. (Using the Unit Production Removal discussed in previous Lecture)

Step 4: Replace each Production $A \rightarrow B_1 \ldots\ldots B_n$ where $n > 2$, with $A \rightarrow B_1 C$ where $C \rightarrow B_2 \ldots\ldots B_n$ Repeat this step for all Productions having two or more Symbols on the right side.

Step 5: If the right side of any Production is in the form $A \rightarrow aB$ where 'a' is a terminal and A and B are non-terminals, then the Production is replaced by $A \rightarrow XB$ and $X \rightarrow a$. Repeat this step for every Production which is of the form $A \rightarrow aB$

Conversion of CFG to Chomsky Normal Form

Convert the following CFG to CNF: P: S→ASA | aB, A→B|S, B→b|∈

1) Since S appears in RHS, we add a new State S' and S'→S is added to the production

P: S'→S, S→ASA|aB, A→B|S, B→b|∈

2) Remove the Null Productions: B→∈ and A→∈ :

After Removing B→∈: P: S'→S, S→ASA|aB|a, A→B|S|∈, B→b

After Removing A→∈: P: S'→S, S→ASA|aB|a|AS|SA|S, A→B|S, B→b

2) Remove the Null Productions: $B \to \in$ and $A \to \in$:

After Removing $B \to \in$: P: $S' \to S,\ S \to ASA|aB|a,\ A \to B|S|\in,\ B \to b$

After Removing $A \to \in$: P: $S' \to S,\ S \to ASA|aB|a|AS|SA|S,\ A \to B|S,\ B \to b$

3) Remove the Unit Productions: $S \to S$, $S' \to S$, $A \to B$ and $A \to S$:

After Removing $S \to S$: P: $S' \to S,\ S \to ASA|aB|a|AS|SA,\ A \to B|S,\ B \to b$

After Removing $S' \to S$: P: $S' \to ASA|aB|a|AS|SA,$
$S \to ASA|aB|a|AS|SA,$
$A \to B|S,\ B \to b$

After Removing $A \to B$: P: $S' \to ASA|aB|a|AS|SA,$
$S \to ASA|aB|a|AS|SA,$
$A \to b|S,\ B \to b$

After Removing $A \to S$: P: $S' \to ASA|aB|a|AS|SA,$
$S \to ASA|aB|a|AS|SA,$
$A \to b|ASA|aB|a|AS|SA,$
$B \to b$

4) Now find out the productions that has more than TWO variables in RHS

$S' \rightarrow ASA$, $S \rightarrow ASA$ and $A \rightarrow ASA$

After removing these, we get: P: $S' \rightarrow AX|aB|a|AS|SA$,
$S \rightarrow AX|aB|a|AS|SA$,
$A \rightarrow b|AX|aB|a|AS|SA$,
$B \rightarrow b$,
$X \rightarrow SA$

5) Now change the productions $S' \rightarrow aB$, $S \rightarrow aB$ and $A \rightarrow aB$

Finally we get: P: $S' \rightarrow AX|YB|a|AS|SA$,
$S \rightarrow AX|YB|a|AS|SA$,
$A \rightarrow b|AX|YB|a|AS|SA$,
$B \rightarrow b$,
$X \rightarrow SA$,
$Y \rightarrow a$

which is the required Chomsky Normal Form for the given CFG

Greibach Normal Form

A CFG is in Greibach Normal Form if the productions are in the following forms:

$$A \rightarrow b$$
$$A \rightarrow bC_1C_2 \ldots\ldots C_n$$

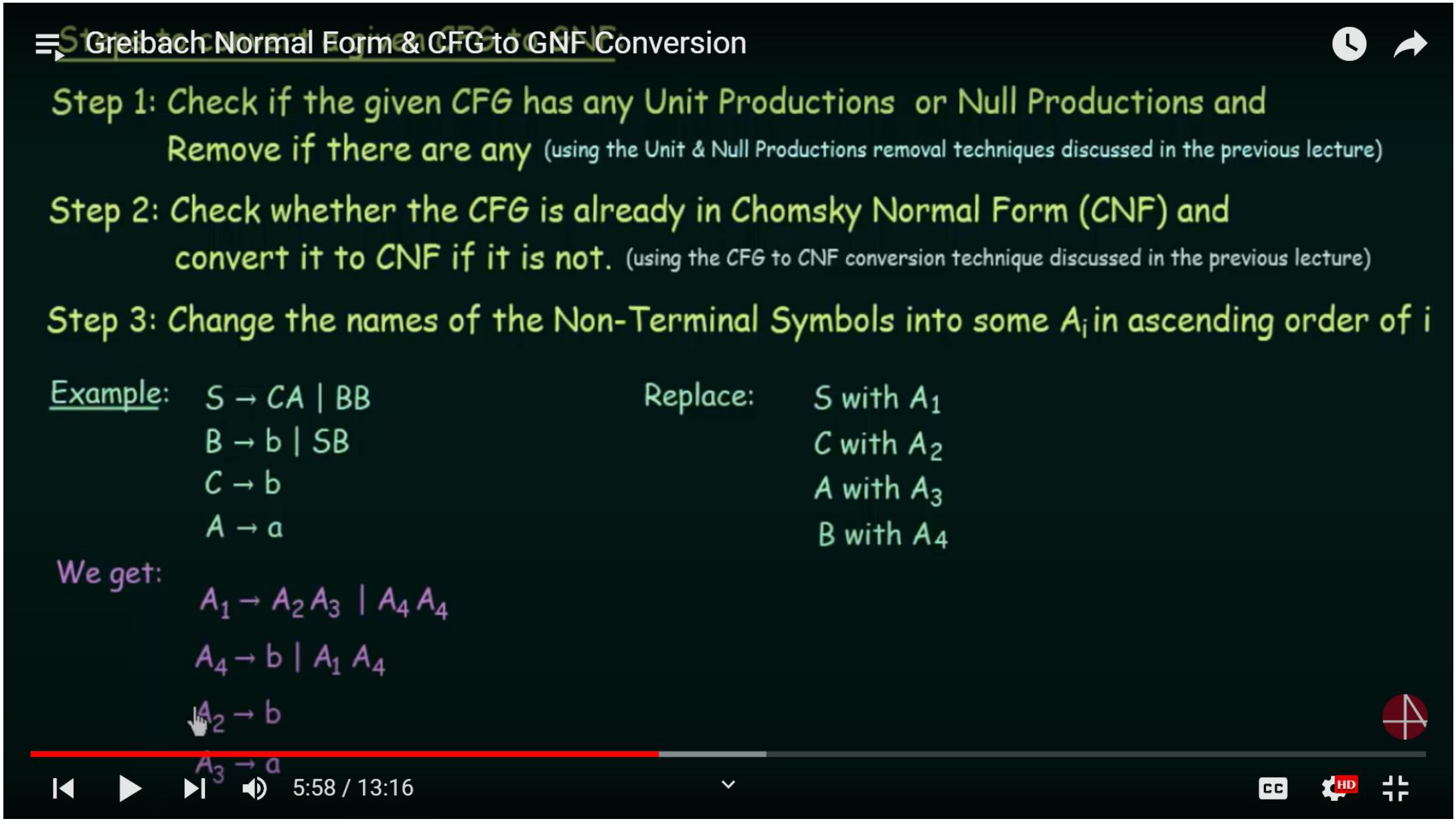
where $A, C_1, \ldots\ldots, C_n$ are Non-Terminals and b is a Terminal

Steps to convert a given CFG to GNF:

Step 1: Check if the given CFG has any Unit Productions or Null Productions and Remove if there are any (using the Unit & Null Productions removal techniques discussed in the previous lecture)

Step 2: Check whether the CFG is already in Chomsky Normal Form (CNF) and convert it to CNF if it is not. (using the CFG to CNF conversion technique discussed in the previous lecture)

Step 3: Change the names of the Non-Terminal Symbols into some A_i in ascending order of i


Greibach Normal Form & CFG to GNF Conversion
Step 1: Check if the given CFG has any Unit Productions or Null Productions and
Remove if there are any (using the Unit & Null Productions removal techniques discussed in the previous lecture)
Step 2: Check whether the CFG is already in Chomsky Normal Form (CNF) and
convert it to CNF if it is not. (using the CFG to CNF conversion technique discussed in the previous lecture)
Step 3: Change the names of the Non-Terminal Symbols into some A_i in ascending order of i
Example:
$S \rightarrow CA \mid BB$
$B \rightarrow b \mid SB$
$C \rightarrow b$
$A \rightarrow a$
Replace:
S with A_1
C with A_2
A with A_3
B with A_4
We get:
$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$
$A_4 \rightarrow b \mid A_1 A_4$
$A_2 \rightarrow b$
$A_3 \rightarrow a$
5:58 / 13:16
CC
HD

$A_2 \rightarrow b$

$A_3 \rightarrow a$

Step 4: Alter the rules so that the Non -Terminals are in ascending order, such that,

If the Production is of the form $A_i \rightarrow A_j\ x$, then,

$i < j$ and should never be $i \geqslant j$

$A_4 \rightarrow b \mid A_1 A_4$

$A_4 \rightarrow b \mid A_2 A_3 A_4 \mid A_4 A_4 A_4$

$A_4 \rightarrow b \mid b A_3 A_4 \mid A_4 A_4 A_4$

↓

Left Recursion

Step 5: Remove Left Recursion

Step 5: Remove Left Recursion

Introduce a New Variable to remove the Left Recursion

$A_4 \rightarrow b \mid b\, A_3\, A_4 \mid A_4\, A_4\, A_4$

$Z \rightarrow A_4\, A_4\, Z \mid A_4\, A_4$

$A_4 \rightarrow b \mid b A_3\, A_4 \mid bZ \mid b\, A_3\, A_4\, Z$

Now the grammar is:

$A_1 \rightarrow A_2\, A_3 \mid A_4\, A_4$

$A_4 \rightarrow b \mid b\, A_3 A_4 \mid bZ \mid b\, A_3\, A_4\, Z$

$Z \rightarrow A_4 A_4 \mid A_4\, A_4\, Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

$| bA_3A_4 | bZ | bA_3A_4Z$

Now the grammar is:

$A_1 \rightarrow A_2A_3 \mid A_4A_4$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow A_4A_4 \mid A_4A_4Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$

$A_1 \rightarrow bA_3 \mid bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4$

$A_4 \rightarrow b \mid bA_3A_4 \mid bZ \mid bA_3A_4Z$

$Z \rightarrow bA_4 \mid bA_3A_4A_4 \mid bZA_4 \mid bA_3A_4ZA_4 \mid$

$bA_4Z \mid bA_3A_4A_4Z \mid bZA_4Z \mid bA_3A_4ZA_4Z$

$A_2 \rightarrow b$

$A_3 \rightarrow a$