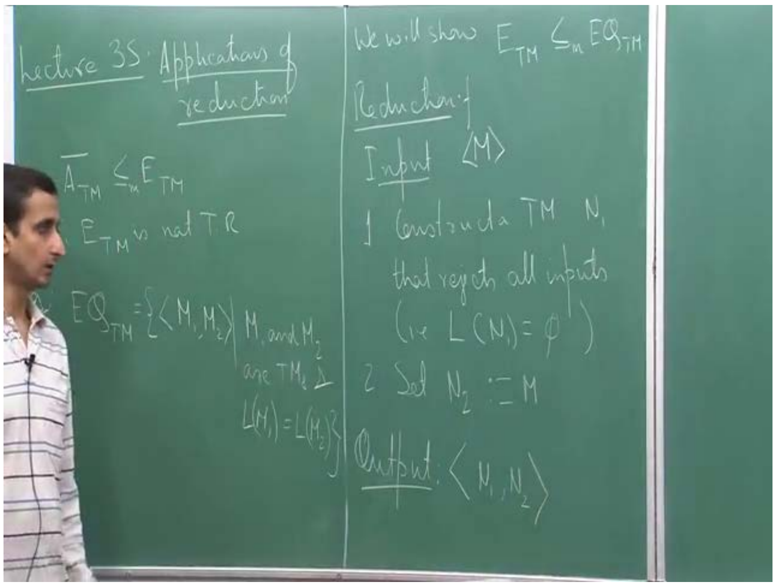
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 35
Applications of Reduction

(Refer Slide Time: 00:2 9)



Lecture 35: Applications of reduction

$\overline{A}_{TM} \le_m E_{TM}$

E_{TM} is not TR

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1$ and M_2 are TMs & $L(M_1) = L(M_2)\}$

We will show $E_{TM} \le_m EQ_{TM}$

Reduction:

Input $\langle M\rangle$

1 Construct a TM N_1 that rejects all inputs (i.e. $L(N_1) = \phi$)

2 Set $N_2 := M$

Output: $\langle N_1, N_2\rangle$

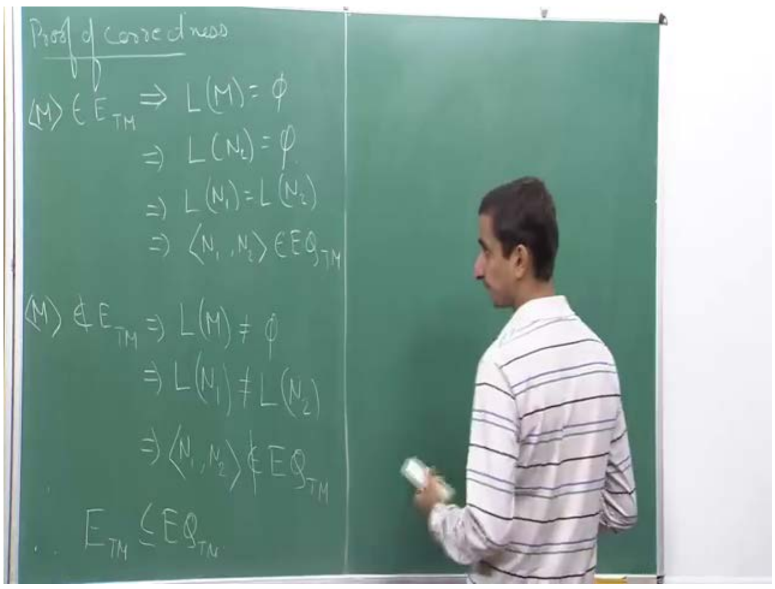
Welcome to the 35 lecture. So, today, we will see some more applications of reduction. So, last time, towards the end, we showed that as an example. We showed that the compliment of A TM reduces to E TM. So, E TM is the language of all encoding of all Turing machines M whose language is empty; and by this, because we know that A TM compliment is not during recognisable, this implies that E TM is not during recognisable. So, we are going to use a similar thing. So, to show this reduction, we constructed computable function F, which took as input and instance of A TM bar which is a machine name and a string w, and it produced an instance of A TM which is an encoding of some machines M such that if M does not accept w then the language of M is empty. And if M accepts w, then the language of M was not empty, so that is what we showed last time.

So, today we will see more examples as to how we can reductions to prove languages are undecidable. So, the first language that we will choose is the language E Q TM. So, E Q TM consists of encodings of two machines M 1 and M 2 such that M 1 and M 2 are

Turing machines and the language of M 1 equals the language of M 2 . So, this is similar to the languages E Q DFA and E Q CFG that we had seen earlier, but in this case will show that this is undecidable. So, how do we prove this? So, what we will show is that E TM reduces to EQTM, so this is what we will show.

So, we will construct a computable function f that will do the following. So, what does f do. So, first f takes as input a machine M let us say. So, first what it will do is that it will construct a Turing machine N 1 that rejects all inputs. So, in other words the language of N 1 is the empty language and it will construct another Turing machine N 2 which will just be the machine M 1. So, I will write it as set N 2 be the machine M; and now it outputs the pair N 1 comma N 2. So, in our reduction the reduction function f what it does is that it takes a machine M and it sets N 2 to be the same machine M. So, it just copies the encoding of M to N 2; and for M, N 1 it constructs a trivial machine that rejects all its input. So, the language N 1 is empty.

(Refer Slide Time: 05:09)



Proof of correctness

$\langle M \rangle \in E_{TM} \Rightarrow L(M) = \phi$

$\Rightarrow L(N_2) = \phi$

$\Rightarrow L(N_1) = L(N_2)$

$\Rightarrow \langle N_1, N_2 \rangle \in EQ_{TM}$

$\langle M \rangle \notin E_{TM} \Rightarrow L(M) \neq \phi$

$\Rightarrow L(N_1) \neq L(N_2)$

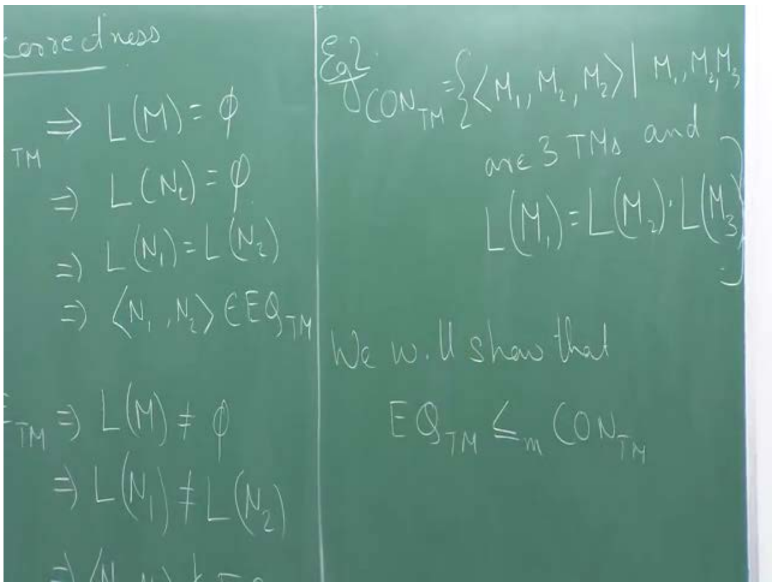
$\Rightarrow \langle N_1, N_2 \rangle \notin EQ_{TM}$

$\therefore E_{TM} \leq EQ_{TM}$

So, now, let us see why this is correct. So, proof of correctness. So, suppose M, the encoding of M belong to E TM. So, what does this imply, so this implies that the language of M is empty. Now this implies because N 2 is nothing but so N 2 here is nothing, but M, so this implies that the language of N 2 is also empty. And we had set the language of N 1 to be empty which implies that the language of N 1 equals the language of N 2, which implies that N 1 comma N 2 is in EQTM.

Now, on the other hand, we have to show both directions. So, it should be so observe that it should show for both for instances which are in the language, and instances which are outside the language it is if and only if condition. So, if M is not an element of E TM, then the language of M is nonempty, which implies because the language of N 2 will also be not-empty. Therefore, it would imply that the language of N 1, which is empty is not equal to the language of N 2 which implies that N 1 comma N 2 is not in E Q TM. So, this completes our proof. Hence we have that E TM reduces to E Q TM.

(Refer Slide Time: 07:44)



orrectness

TM $\Rightarrow L(M) = \phi$

$\Rightarrow L(N_2) = \phi$

$\Rightarrow L(N_1) = L(N_2)$

$\Rightarrow \langle N_1, N_2 \rangle \in EQ_{TM}$

TM $\Rightarrow L(M) \neq \phi$

$\Rightarrow L(N_1) \neq L(N_2)$

Eg 2:

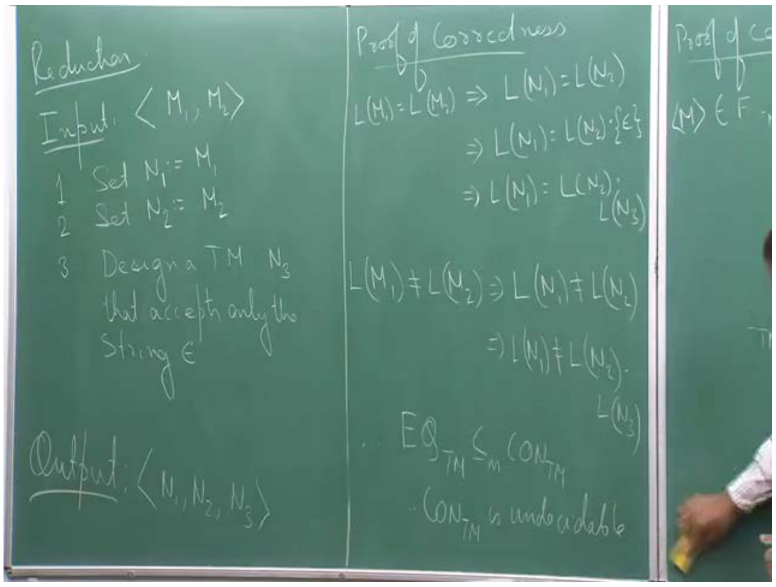
$CON_{TM} = \{ \langle M_1, M_2, M_3 \rangle \mid M_1, M_2, M_3$ are 3 TMs and $L(M_1) = L(M_2) \cdot L(M_3) \}$

We will show that

$EQ_{TM} \leq_m CON_{TM}$

So, let us look at another simple example. So, consider a language let us call to CON TM So, CON stands for concatenate, so it consists of encodings of 3 machines M 1, M 2 and M 3. So, M 1 M 2 and M 3 are three Turing machines. And the language of M 1 is equal to the language of M 2 concatenated with the language of M 3. Let us say. So, I am given 3 machines such that they have this particular property and I want to proof that CON TM is undecidable. So, what we will show is, so now that we have proven E Q TM is undecidable we will show that E Q TM reduces to CON TM. So, how do we proof this?

(Refer Slide Time: 09:34)



Reduction:

Input: $\langle M_1, M_2 \rangle$

1 Set $N_1 := M_1$

2 Set $N_2 := M_2$

3 Design a TM N_3 that accepts only the String ϵ

Output: $\langle N_1, N_2, N_3 \rangle$

Proof of Correctness

$L(M_1) = L(M_2) \Rightarrow L(N_1) = L(N_2)$

$\Rightarrow L(N_1) = L(N_2)\cdot\{\epsilon\}$

$\Rightarrow L(N_1) = L(N_2)\cdot L(N_3)$

$L(M_1) \neq L(M_2) \Rightarrow L(N_1) \neq L(N_2)$

$\Rightarrow L(N_1) \neq L(N_2)\cdot L(N_3)$

$\therefore EQ_{TM} \leq_m CON_{TM}$

$\therefore CON_{TM}$ is undecidable

Proof of Co

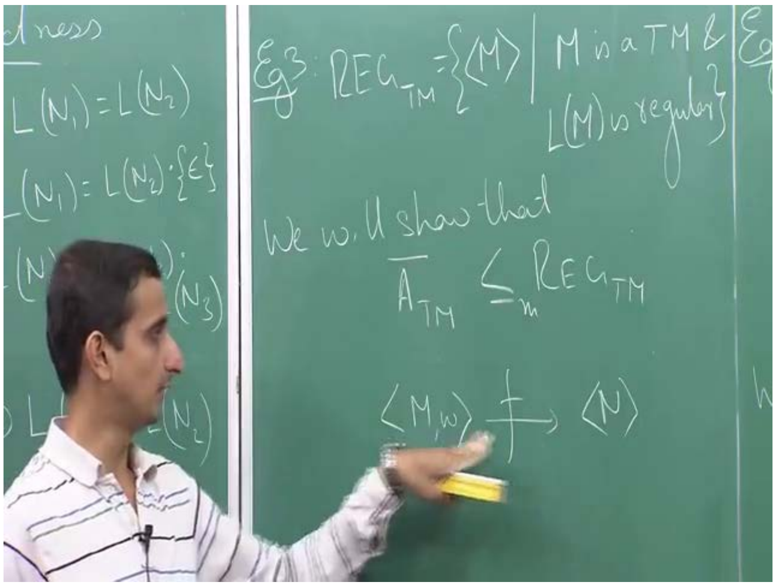
$\langle M \rangle \in F$

So, let us construct out reduction. So, we construct a function f that takes as input. So, what do we want to reduce? So, we want to reduce E Q TM to CON TM. So, our in input should be an instance of E Q TM. So, I take as input and instance of E Q TM. So, an instance of E Q TM is an encoding of two Turing machines. So, let us say I take the machine M 1 and M 2 as input. And what I want to output let me write it down is an instance of CON TM, which is an encoding of three Turing machines. So, I will have N 1 N 2 and N 3. So, what do we do? So, first we set N 1 as equal to M 1, so the machine N 1 is nothing but the machines M 1. Set N 2 to be the machines M 2, once again they are the same machine. And for M 3, what we will do is that, so design a Turing machine N 3 that accepts only the string epsilon. So, N 3 accepts only one string which is epsilon. So, it checks what its input is. So, if input is some non-empty string, it will reject; only if its input is the empty string epsilon, it is going to accept; otherwise, it will always rejects.

Now let us see why our reduction works correctly. So, suppose M 1 comma M 2 is an instance of EQTM. What does that mean, so it means that L of M 1 equals L of M 2. So, L of M 1 equals L of M 2 then what we have is that L of N 1 equals L of N 2, because M N 1 is M 1 and N 2 is M 2. Now I use an identity from regular expression. So, remember that we had an identity that suppose if I have a language L, so if I look at the concatenation of L with the empty string, so if I concatenate L with the language which contains only the empty string, then I get the language L itself. Because the empty string concatenate with any string gives back the same string, hence I get the language L. So, this implies that this is nothing, but L of N 1, which is equal to L of N 2 concatenated with only the string epsilon. But by my construction this is nothing but L of N 1 equals L of N 2 concatenated with L of N 3, because that is how I designed N 3 to contain only the empty string. So, this proofs that N 1, N 2, N 3 is contained in CON TM.

On the other hand, if L M 1 is not equal to L M 2, then L of N 1 is not equal to L of N 2, which implies that L of N 1 is not equal to L of N 2 concatenated with L of N 3, which is nothing but the empty string. So, L of n 3 concatenated with L of N 2 is just L of n 2. So, therefore, we have this. So, this proves that EQTM reduces to con tm therefore, con tm is undecidable.

(Refer Slide Time: 16:02)

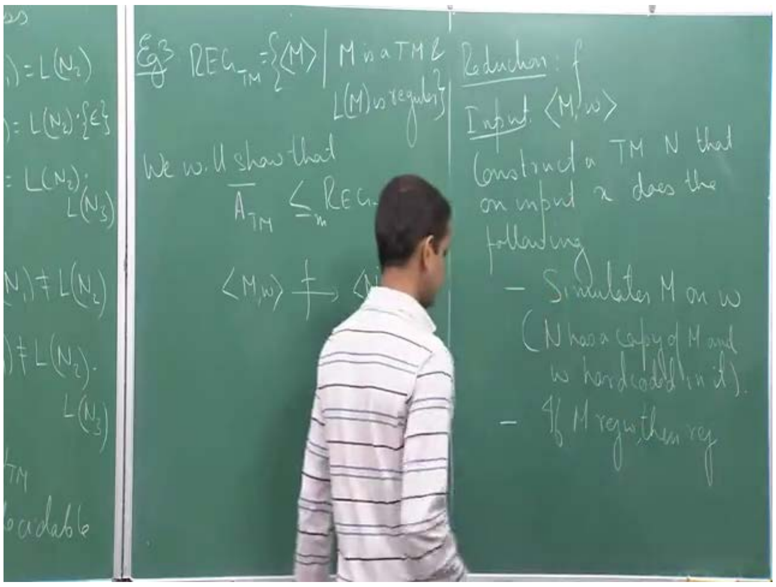


$L(N_1) = L(N_2)$
$L(N_1) = L(N_2) \cdot \{\epsilon\}$
Eg3: $REG_{TM} = \{\langle M \rangle \mid M$ is a TM & $L(M)$ is regular$\}$
We will show that
$\overline{A}_{TM} \leq_m REG_{TM}$
$\langle M, w \rangle \xrightarrow{f} \langle N \rangle$

So, now, let us look at the last example of today. So, now, we look at a slightly more non-trivial example, it is little bit more involved in the first two examples. So, suppose let us define this language REG TM as encodings of Turing machines M. So, M is a Turing machine and the language of M is regular. So, a language of a Turing machine can be anything I mean it can also be non regular language, but let us say that of course, I can have Turing machine, for example, which accept the empty set or Turing machine that accept sigma star or some other regular language. So, I do have certain Turing machines that accept regular languages. So, consider the class or consider the language of encodings of all those Turing machines, whose language is regular.

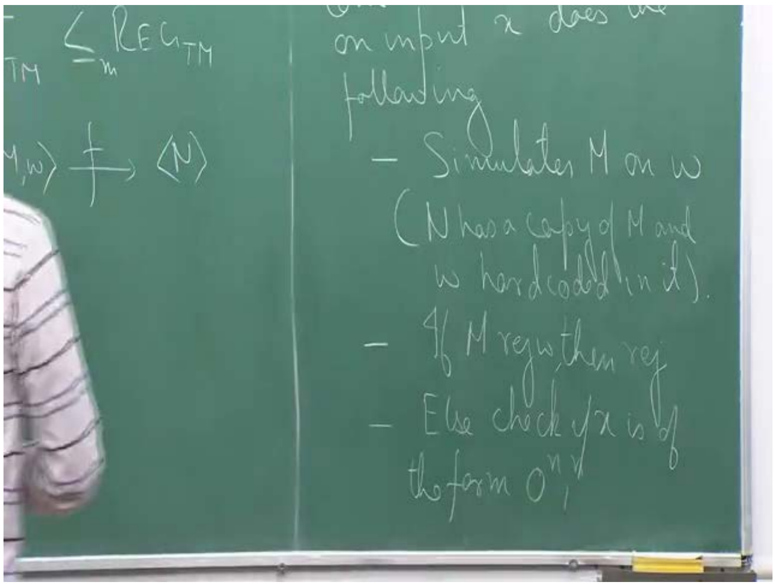
So, what we will prove is that A TM compliment reduces to REG TM. And this would show that REG TM is undecidable in particular it is not even Turing recognisable. So, once again, we need to design computable function f. So, what is an instance, an instance of A TM bar will be a pair of the following M comma w; and an instance of REG TM is a single machine let say n. So, given M comma w, I want to map it to a machine N; such that if M does not accept w then this guy is regular; and if N accepts w, then this guy is not regular.

(Refer Slide Time: 18:16)


Eg3 $REG_{TM} = \{\langle M\rangle \mid M$ is a TM & $L(M)$ is regular$\}$
We will show that
$\overline{A_{TM}} \le_m REG$
$\langle M,w\rangle \xrightarrow{f}$
Reduction: f
Input: $\langle M,w\rangle$
Construct a TM N that on input x does the following
– Simulates M on w
(N has a copy of M and w hardcoded in it).
– If M rej. then rej.

So, what is our reduction going to be. So, we take as input a machine M comma w. So, what we do now is that we construct a machine or we write as Turing machine, so construct a Turing machine N that on input x does the following. So, what does N do. So, N of course has copy of M and w hardcoded in it. So, N simulates M on w. So, N has a copy of M and w hardcoded in it. So, it simulates M on w. If M rejects, then reject. So, what the machine N does is if M rejects w then it rejects, M rejects w then reject.

(Refer Slide Time: 2 0:18)



$\leq_m REG_{TM}$

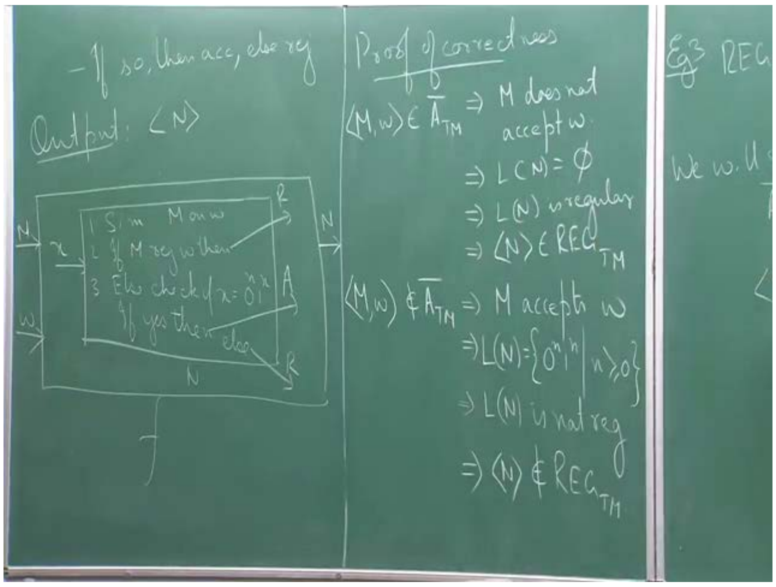
$\langle M,w\rangle \xrightarrow{f} \langle N\rangle$

on input x does the following

- Simulates M on w (N has a copy of M and w hardcoded in it).
- If M rej w, then rej
- Else check if x is of the form $0^n 1^n$

Otherwise, else check if x is of the form 0 to the power n 1 to the power.

(Refer Slide Time: 2 0:47)



- If so, then acc, else rej

Output: $\langle N\rangle$

Proof of correctness

$\langle M,w\rangle \in \overline{A}_{TM} \Rightarrow$ M does not accept w.

$\Rightarrow L(N) = \phi$

$\Rightarrow L(N)$ is regular

$\Rightarrow \langle N\rangle \in REG_{TM}$

$\langle M,w\rangle \notin \overline{A}_{TM} \Rightarrow$ M accepts w

$\Rightarrow L(N) = \{0^n 1^n \mid n \geq 0\}$

$\Rightarrow L(N)$ is not reg

$\Rightarrow \langle N\rangle \notin REG_{TM}$

Eg3 REG

We will

And. So, let me continue here. If so then accept, else reject. So, what are we doing essentially, so let us go through it. First, I simulate M on w, so they are hardcoded into the description of N. So, what the machine N does at the beginning is it simulates M on w. If it loops forever then of course, N will also look forever; it will never stop. Otherwise, it checks the answer. So, if M rejects w, then N will immediately reject; else, so if M accepts w, it checks whether the input of N, so the input of N was x, so it checks whether x is of the form 0 to the power n 1 to the power n. So, why do we take this. So, we take a string of the form 0 to the power n 1 to the power n, because we know that this is, so strings of this form gives as a non regular language, so that is the first non regular language that we saw. So, that is why we check if the string as this particular form. If it has this particular form, then we accept, else reject. And our output is the machine N.

So, if I want to give a box diagrammatic picture of what is happening. So, this is my reduction f. So, what is a reduction doing, it is taking two things as input, it takes a machine M and it takes a string w, and it outputs another machine N. Now how does it define N, so this machine N is defined as follows, so this is the machine N. It takes as input x. So, first it simulates M on w. So, if M rejects w, then reject; else, check if x is of the form 0 to the power n 1 to the power n; if yes, then it accepts; else, it rejects. So, it rejecting in these two cases, it is accepting here. So, this is the design of the machine and finally, outputs the machine.

So, now, let us try to analyse the proof of correctness of this reduction. So, suppose M comma w does not belong to A TM or in other words let say that M comma w belongs to A TM bar. So, this means that M does not accept w. So, if M does not accept w, two things can happen. In the first step, either it loops on w forever in which case I do not even go to the second step or it reject w, in which case the machine N rejects x. So, in either two cases, the machine N observes that it does not accept any x. So, all x is our either getting rejected or the machine N loops forever on x. So, this implies that the language of N is the empty set which implies that the language of N is regular, because the empty set is a regular language, which implies that N belongs to REG TM.

On the other hand, if M comma w does not belong to A TM bar, it means that M accepts w. So, in this case what happens, so if M accepts w then we go to step 3, we check if x as this form if x is of the form 0 to the power n 1 to the power n; if so, then we accept else we reject. So, what will be the language of N, in this case? So, the language of N in this case is going to be all strings, which have the form o to the power n 1 to the power n. So, in this case the language of N is set of all strings of the form 0 to the power n 1 to the power n for n greater than or equal to 0. So, this implies that the language of N and we know that this is a non-regular language. So, the language of N is not regular which implies that N does not belong to REG TM. So, this completes our proof.

So, now, similar idea can also be used to show that CFL TM is undecidable. So, CFL TM consists of encodings of all Turing machines, whose language is CFL or a context free language. So, in a similar manner, I can construct a language of strings which is not a context free language; and on the other hand let say I construct context free language let say the empty language and we will be done. So, what we will see next time, in our next lecture is a generalization of this technique. So, generalization is known as Rice’s theorem. So, in our next lecture, we first understand what how to define a generalization of this theme, we will look at the statement of Rice’s theorem, and then we will see how to prove it.

Thank you.