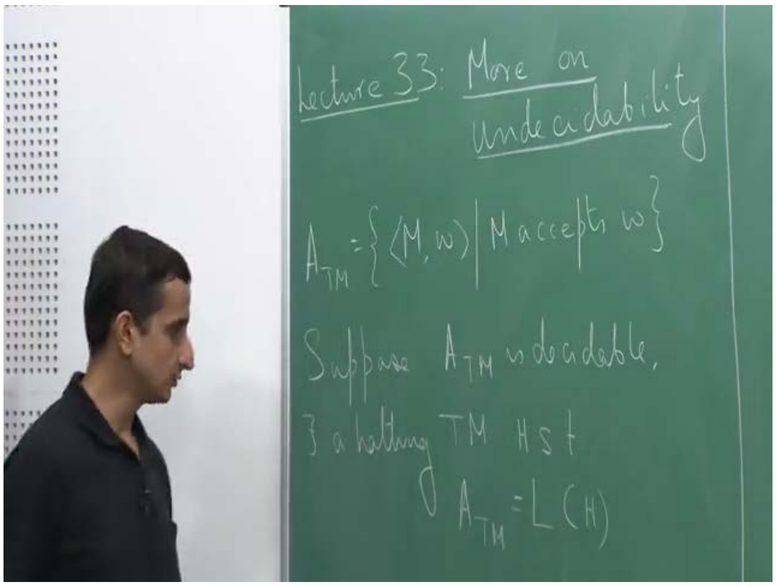
Theory of Computation

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

Lecture – 33
More on Undecidability

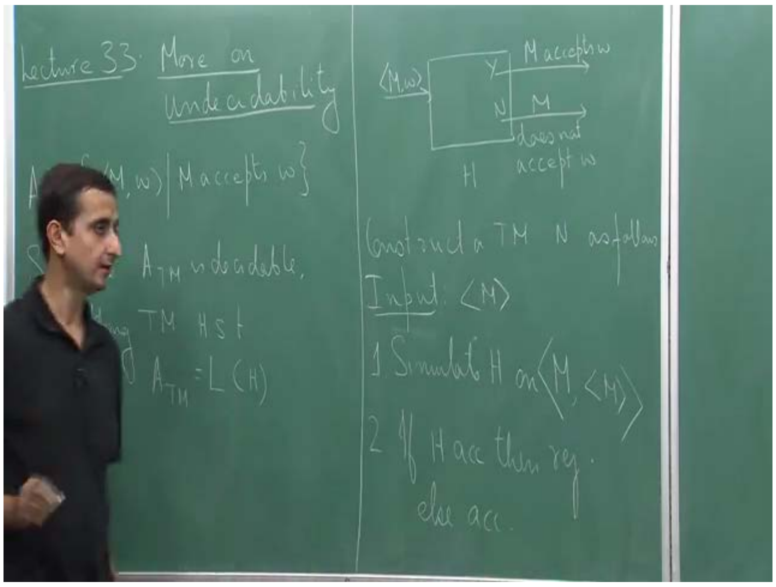
Welcome to the 33 lecture of this course. So, today we are going to continue our discussion on undecidability.

(Refer Slide Time: 00:25)


Lecture 33: More on Undecidability
$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$
Suppose A_{TM} is decidable.
$\exists$ a halting TM H s.t.
$A_{TM} = L(H)$

In our last lecture, we had introduced the problem A TM; A TM is defined as all tuples M w such that M accepts w. And we had shown that the problem ATM is Turing recognizable. So, what we will prove today is that ATM is undecidable. So, suppose ATM is decidable then there exists a halting Turing machine let say H such that ATM is equal to L of H.

(Refer Slide Time: 01:30)

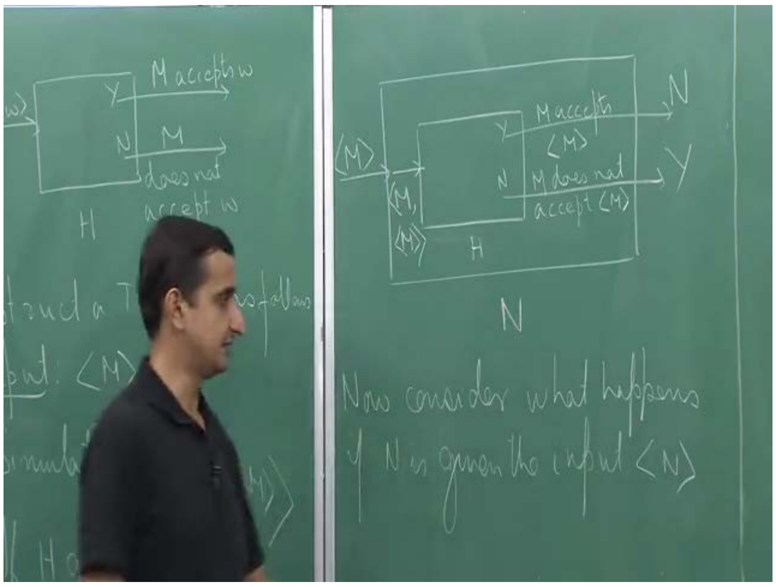

Lecture 33 · More on Undecidability
$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$
A_{TM} undecidable.
TM H s.t.
$A_{TM} = L(H)$
$\langle M, w\rangle$
Y
M accepts w
N
M does not accept w
H
Construct a TM N as follows
Input: $\langle M\rangle$
1 Simulate H on $\langle M, \langle M\rangle\rangle$
2 If H acc then rej.
else acc.

So, what essentially this means is that. So, if I think of the Turing machine as a box. So, this is my Turing machine H then given as input M comma w together, it will say yes if M accepts w, and it will say no if M does not accept w. And note that H is a halting Turing machine which means that for every pair of the form M comma w, it must give an answer either yes or no.

Now using H, we will construct another machine. So, construct a Turing machine N as follows. So, what the Turing machine N does is that it takes as input encoding of a machine M, and then what it does is that it will simulate H on M comma the encoding of M. Let us try to understand the meaning of this statement. So, what essentially this means is that. So, we are designing this new machine N. So, once again think of it as a box or you may also want to think of it as a computer program.

So, here I want to write a program that simulates some other program some other Turing machine. So, you can think of this as a function that the machine N is calling. So, this H is a Turing machine which means it has some finite description. So, always the description of H can be hardwired into the description of N. So, N has a description of H hardwired onto it. But N does is that in the first step it looks at the encoding of M, so it treats it as a string, and it also treats it as a machine and what it does is that it will simulate H on the pair M comma the string corresponding to M. So, if H accepts then reject, else accept.

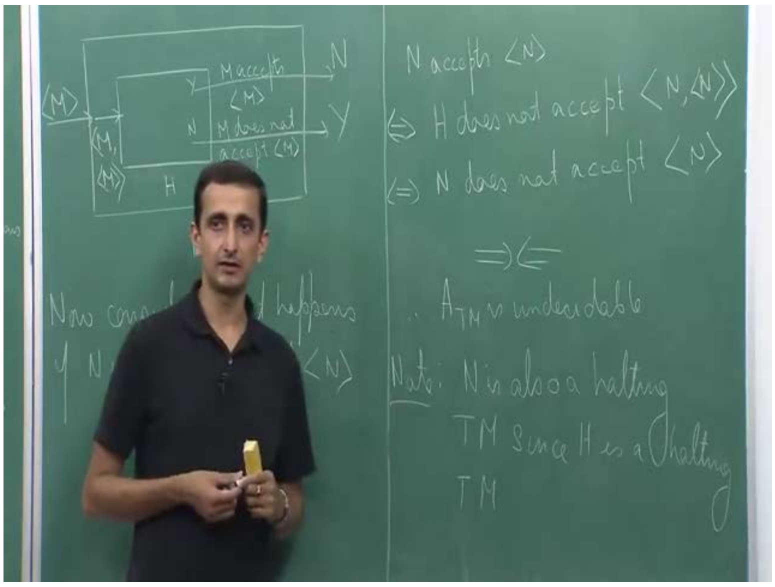
(Refer Slide Time: 04:33)



Let me just represent this in this box format. So, our machine N has the following. So, it takes as input let say encoding of a machine M and what it does is that inside itself, it has the machine H hardwired I mean a encoding of the machine H. So, after taking M it feeds M comma then coding of M to the machine H. Now H can do two things right. So, either H can say yes, so if H says when does H says yes. So, H will say yes, if M accepts the encoding of M; and H will say no, if M does not accept the encoding of m.

But H is a halting Turing machine which means that H always gives a yes no answer. So, if H gives a answer yes, I will give an answer no; and if H gives no, I will give an answer yes, the machine n. So, if H says yes, the machine N says no; and if H says no, the machine N says yes. So, this is actually the entire description of the Turing machine N. Now observe what happens. Now consider what happens if N is given the input which is actually an encoding of N itself. So, N as a Turing machine can take the encoding of any machine as input. So, consider the case when it takes the encoding of itself as an input.

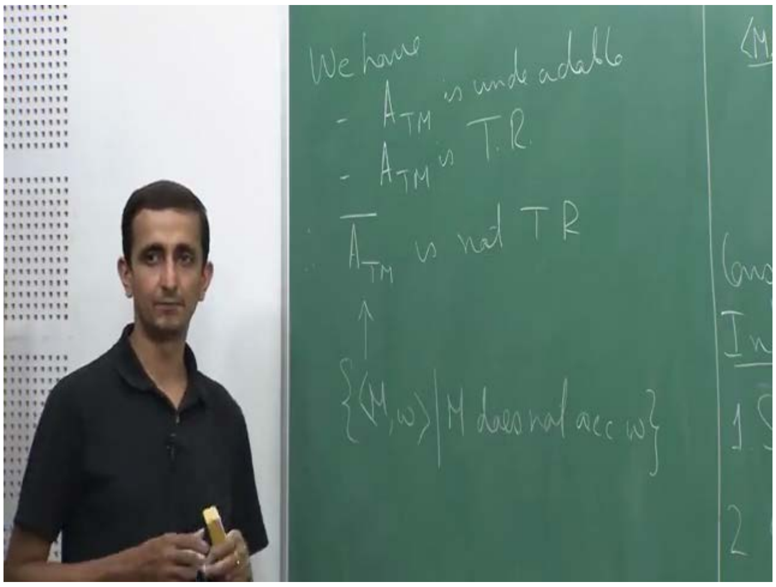
(Refer Slide Time: 06:56)


⟨M⟩
⟨M, ⟨M⟩⟩
H
Y
N
M accepts ⟨M⟩
M does not accept ⟨M⟩
N
Y
N accepts ⟨N⟩
⟺ H does not accept ⟨N, ⟨N⟩⟩
⟺ N does not accept ⟨N⟩
⇒⇐
∴ A_TM is undecidable
Note: N is also a halting TM since H is a halting TM
Now consider what happens if N ... ⟨N⟩

So, if N accepts N if and only if, so when does the machine N accept N. The machine N will accept N in this case, when the machine H does not accept M comma M right if and only if H does not accept N comma N. And when does H not accept N comma N. So, what is the machine H, let us go back. The machine H is a machine which takes as input a machine and a string and it does not accept the input M comma w, if M does not accept w. So, therefore, here if I come back H do not accept N comma N if and only if N does not accept the string corresponding to the encoding of N.

So, we have that N accepts the encoding of N if and only if N does not accept the encoding of N. And this is a contradiction. So, therefore, A TM is undecidable. So, just a small note that I had briefly mentioned, so note that N is also a halting Turing machine, since H is a halting Turing machine, because it can never go onto a infinite loop because it is only simulating H. So, this proves that the language A TM is undecidable.

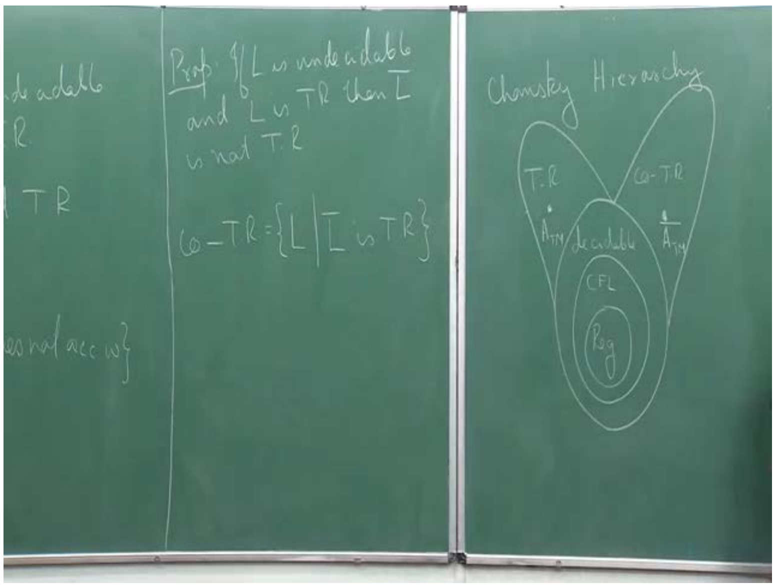
(Refer Slide Time: 09:35)


We have
- A_{TM} is undecidable
- A_{TM} is T.R.
$\therefore \overline{A_{TM}}$ is not T R
$\{\langle M,w\rangle \mid M \text{ does not acc } w\}$

Now what does essentially this mean? So, we have A TM. So, we have that A TM is undecidable. And in our last lecture, we had shown that A TM is also Turing recognizable. So, what does this tell us about A TM compliment or A TM bar? Therefore, A TM compliment is not Turing recognizable. So, if I just want to write it down, so ATM bar is the set of all pairs M w such that M does not accept w, here we have another language that is not Turing recognizable. Why is this I mean why can we conclude that A TM bar is not Turing recognizable.

So, recall a theorem that we did a couple of lectures back, where we said that a language is decidable if and only if it is both Turing recognizable and its compliment is Turing recognizable. So, here what is turns out is that I have A TM which is Turing recognizable, now if the compliment of A TM is also Turing recognizable that could immediately imply that ATM is decidable, but we have proved while back that A TM is undecidable hence A TM compliment cannot be Turing recognizable.

(Refer Slide Time: 11:28)



Prop: If L is undecidable and L is TR then $\bar{L}$ is not TR

$co-TR = \{L \mid \bar{L} \text{ is } TR\}$

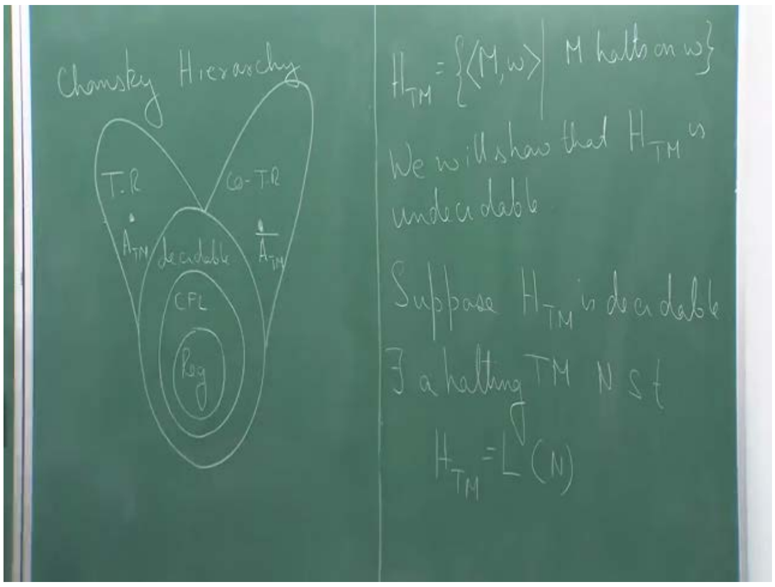
Let me just write this as a proposition or this is more like a corollary, but so if L is undecidable and L is Turing recognizable then L compliment is not Turing recognizable. So, this is a very important thing that is used on several occasions to argue that certain language is a not Turing recognizable.

Now with the picture that we currently have let us go back to the Chomsky Hierarchy recall the Venn diagram that we had the egg shaped Venn diagram that we had drawn few lectures back which represented the relation between the various classes. Let us go back to that Venn diagram and let us try to understand the relation between the old classes versus the new. So, we have the set of regular languages this is the smallest say then we have the set of context free languages, then we have the set of decidable languages. So, they form a superset of the set of context free languages.

And now we have Turing recognizable languages and what are called co Turing recognizable languages. So, these are Turing recognizable languages, and we have what are called co Turing recognizable languages. Let me define the co Turing recognizable languages. It consist of all languages L such that L bar is Turing recognizable that is all, so all the languages whose compliment is a Turing recognizable language. So, clearly so there cannot be any problem which is in the intersection of Turing recognizable and co Turing recognizable languages and which are not decidable. So, by definition the intersection of T R and co T R is decidable.

Let us put some class. So, we have the problem A TM. So, this is the problem A TM. So, ATM belongs to Turing recognizable, but it is not decidable. And therefore, we have A TM bar which is co Turing recognizable by definition, but it is not Turing recognizable, and it is not decidable certainly. It cannot be decidable by because of course; the classes of decidable languages are closed under compliment. So, this is the current picture of the Chomsky hierarchy that we have.

(Refer Slide Time: 15:44)


Chomsky Hierarchy
T.R
Co-T.R
A_{TM}
decidable
$\overline{A_{TM}}$
CFL
Reg

$H_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$

We will show that H_{TM} is undecidable

Suppose H_{TM} is decidable

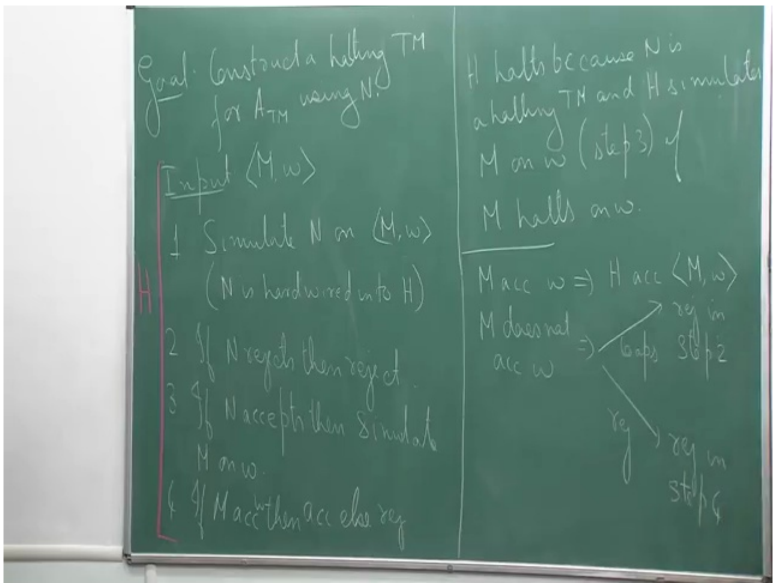
$\exists$ a halting TM N s.t

$H_{TM} = L(N)$

Now, let us look at another problem - another undecidable problem. So, this is popularly known as the halting problem. So, we define it ash tm. So, H TM consists of encodings of the form M comma w such that A TM halts on w. So, we will show that H TM is also undecidable. So, how do we prove this? So, we will prove this by somehow using the fact that. In fact, by using entirely using the fact ATM is actually undecidable. So, suppose for the sake of contradiction, H TM is decidable, so what does that mean if H tm is decidable it means that there exists a halting Turing machine.

Let us give it a name N such that H TM is equal to L of N. So, N accepts the language H TM. Now what we will do is that our goal is to use the machine N, and construct a decider for A TM. In other words, I want to construct a halting Turing machine that would accept A TM, but that we have just proven a little while back that it is not possible ATM is undecidable, which will contradict the fact that H TM is decidable.

(Refer Slide Time: 18:23)



Goal: Construct a halting TM for A_{TM} using N.

H
Input $\langle M, w\rangle$
1 Simulate N on $\langle M, w\rangle$
(N is hardwired into H)
2 If N rejects then reject.
3 If N accepts then simulate M on w.
4 If M acc w then acc else rej

H halts because N is a halting TM and H simulates M on w (step 3) if M halts on w.

M acc w $\Rightarrow$ H acc $\langle M, w\rangle$
M does not acc w $\Rightarrow$ loops $\rightarrow$ rej in step 2
rej $\rightarrow$ rej in step 4

So, our goal is to construct a halting Turing machine for ATM using N. So, what do we do? So, here is the construction. So, we take as input M comma w, and our goal is accept if M accepts w, otherwise reject. So, first step what we do is that we simulate H or N on M comma w.

So, note that once again H is hardwired into this machine. Let us give this machine a name. Let us call this machine as H, so sorry not H, here N. So, N is hardwired into H. So, this machine N that we assumed for H TM that is hardwired into the code of H or into the transition function of H; so what I can do is that I can simulate N on the input itself on M w. So, if N rejects, then reject; if N accepts then simulate M on w. So, once again I have of course, M and w as input. So, I can of course, simulate M on w. So, I simulate M on w, if M accepts then accept, else reject that is all.

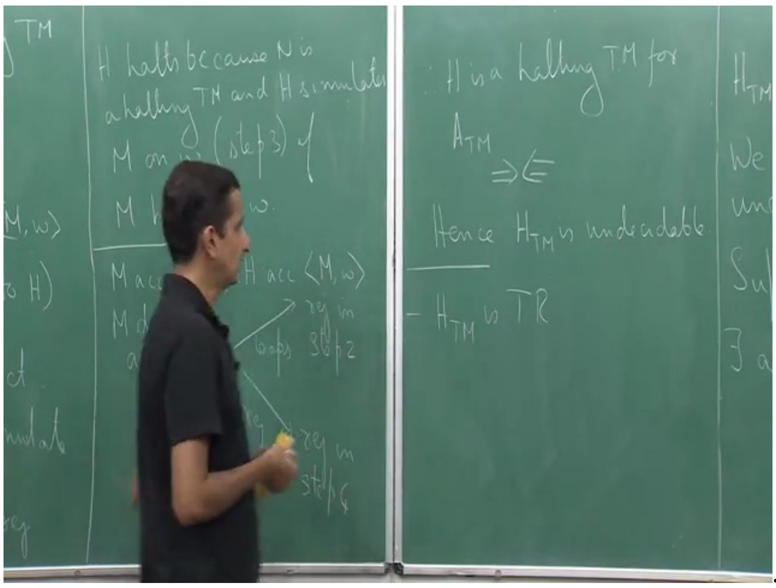
Now let us try to prove the correctness of this algorithm. So, first of all, what I want to argue is that H is a halting Turing machine; it never goes into an infinite loop at any point. So, why is that, so look at the first step, so in the first step what it is doing is it is just simulating N on M comma w, because N is a halting Turing machine, the first we will always halt with an answer.

Now in the second step, if N rejects, so when does N reject. So, N rejects M comma w if M does not halt on w. If M halts on w, then N will accept. So, if N rejects then I will immediately reject. Now if N does not reject, which means that if N accepts which means that M halts on w, I will go add and simulate M on w. I know that M halts on w in this case. Now, if M accepts then accept, else reject. So, this shows that this Turing machine H would always halt.

So, H halts, because N is a halting Turing machine and H simulates M on w that is basically step three if M halts on w. So, if it has the guarantee that M halts on w only then it is simulating. So, this proves that H is a halting Turing machine. Now, let us look at correctness. So, suppose here M accepts w. So, if M accepts w then in the first step, what we get is that the simulation of N on M w would always accept, because if it accepts then it must also halt. Then we go into step 2, it does not go into the condition of step 2, so N accepts, so therefore, it simulates M on w. M accepts w here and therefore, the machine H accepts. So, if M accepts w then H will accept M comma w.

On the other hand, if M does not accept w, what is happening? So, if M does not accept w, what we have is that either it goes into an infinite loop on w. So, it either loops or it rejects. So, if it loops, then it will get rejected in step 2. So, then it basically gets rejected in step 2, because N is a decider for the halting Turing machine. On the other hand, if it actually does not loop, if it actually rejects, then it gets rejected in step 4, because after simulating M on w, we have that it rejects. So, therefore, if M does not accept w, then in both these cases H will reject w.

(Refer Slide Time: 26:01)


H halts because N is a halting TM and H simulates M on w (step 3) if M halts on w.
M acc w ⇒ H acc ⟨M,w⟩
rej in step 2
loops
rej in step 4
∴ H is a halting TM for A_{TM}
⇒⇐
Hence H_{TM} is undecidable.
- H_{TM} is TR
.

Therefore, H is a halting Turing machine for A TM, but this is a contradiction hence H TM is undecidable this is what we assumed at the beginning. So, this completes our proof. Now, by similar argument, I mean one can show that H TM is also Turing recognizable. Why is that so, I mean I can just simulate M on w given M on w I create a machine which simulates M on w; if it accepts or rejects I would accept because that means, that it halts. And if does not halt then it just go on forever, I do not accept. So, this proves that H TM is Turing recognizable which again by our earlier proposition implies that the compliment of H TM is not Turing recognizable.

So, I will stop here today. So, what we will see next time is a formulization of this idea. So, what we did in this idea to show that H TM is undecidable is we used to show that H TM is undecidable. We assume that H tm is actually decidable and we used it as a black-box to construct an algorithm for a known undecidable problem. So, we used it as a black-box. So, this idea of using one problem as a black-box to solve another problem is what is known as reduction in computer science. So, again this is a very important concept. So, in our next lecture, we will formalize this idea of reduction, what does it mean to reduce 1 problem to another, and we will see how it can be used to prove more undecidable problems.

Thank you.