

**Theory of Computation**  
**Professor Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**  
**Lecture 42**  
**Halting Problem**

(Refer Slide Time: 0:17)

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \}$



One attempt:

$U =$  On input  $\langle M, w \rangle$ .  $M$  is a TM and  $w$  is a string.

1. Simulate  $M$  on  $w$ .
2. If  $M$  accepts, accept.  
If  $M$  rejects, reject. *Not happening*

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . *A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.*

$A_{TM}$  is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop



Hello and welcome to lecture number 37 of the course Theory of Computation. In the previous lectures, we saw some decidable languages, then we started seeing the theory of countable sets and uncountable sets. And using that we were able to see that certain sets are countable and certain sets are not countable. In particular, we saw that the set of all Turing machines is a countable set because it is equivalent to the or a subset of the set of all finite strings whereas, the set of all languages is an uncountable set.

Hence, together this implies that there is a language that is not Turing recognizable which is also a language that is not decidable because, if a language is decidable that means it is Turing recognizable as well. So, if it is not getting recognizable it means it is not decided less well, but we did not say or we did not see any specific language that was not Turing recognizable. So, we just said that there are languages which are not Turing recognizable, it was an existential sort of proof. It just showed that this is small and this is high in number then the set of Turing machines a small, set of languages high. So, there has to be languages that cannot be captured by Turing machines. So, it was an existential proof.

In this lecture, we are going to see a specific language which is not decidable. So, we will first see that language and using that we will now we later be able to see other languages also which are not decidable. So, the first language that we will see which is undecidable is A T M. It is the acceptance problem of a Turing machine. So, given a Turing Machine M and a string w, the question is, does M accept w or not?

So, it seems like a very simple straightforward question, but this happens to be an undecidable problem. So, one natural attempt I am trying to show is that this is decidable. So, I told you upfront that it is undecidable, but one attempt at trying to show that this is decidable would be to do the following. So, let us say U is a proposed decider, the U that is mentioned here. And what does U do? U takes M takes the description of M and runs the Turing Machine M on w. It simulates the running of M on w. If M accepts U accepts the pair M w, if M rejects w then U rejects the pair M comma w. But so, this seems like it has to work.

But there is a small issue here. The thing is, if M accepts w, we will accept the pair M comma w because if M accepts, if M rejects w, we will reject the pair M comma w. But, there is a third possibility that M may loop on w if U is a decider for A T M if U is a decider, even for the case where M loops on w, U have to reject the pair M comma w. So, this is the main issue. So, this is only a recognizer.

So, A deciders must reject M comma w when sorry when M loops on w as well so, A decider must also reject M comma w when M loops on w as well. That is not happening here. This is not happening here. So, U does not reject M comma W when M loops on w because U will just continue the simulation, M is running on w and it is looping so, it just keeps running forever and U will also keep running forever . It will neither accept nor reject it.

So, what we want is something that accepts, if M accepts w and rejects when M does not accept w. So, we want it to reject in both cases, when M rejects w, as well as M loops on w. And this turns out to be extremely tricky.

(Refer Slide Time: 5:26)



1. Simulate  $M$  on  $w$ .  
2. If  $M$  accepts, accept.  
If  $M$  rejects, reject. ↑ Not happening

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

A TM is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ , it can reject  $w$ . This is the hardest part.

Real Halting Problem  
 $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

$HALT_{TM}$  is also undecidable. We will also refer to  $HALT_{TM}$  as the "halting problem."





1. Simulate  $M$  on  $w$ .  
2. If  $M$  accepts, accept.  
If  $M$  rejects, reject. ↑ Not happening

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

A TM is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ , it can reject  $w$ . This is the hardest part.

Real Halting Problem  
 $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

$HALT_{TM}$  is also undecidable. We will also refer to  $HALT_{TM}$  as the "halting problem."



This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

$A_{TM}$  is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ ,  $U$  can reject  $w$ . This is the hardest part.

### Real Halting Problem

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$$

$HALT_{TM}$  is also undecidable. We will also refer to  $A_{TM}$  as the "halting problem."

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$



This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

$A_{TM}$  is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ ,  $U$  can reject  $w$ . This is the hardest part.

### Real Halting Problem

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$$

$HALT_{TM}$  is also undecidable. We will also refer to  $A_{TM}$  as the "halting problem."

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$



One attempt:

$U =$  On input  $\langle M, w \rangle$ .  $M$  is a TM and  $w$  is a string.

1. Simulate  $M$  on  $w$ .
2. If  $M$  accepts, accept.  
If  $M$  rejects, reject.


Not happening

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

$A_{TM}$  is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ ,  $U$  can reject  $w$ . This is the hardest part.

### Real Halting Problem





Halting Problem is undecidable

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \}$


One attempt:

$U = \text{On input } \langle M, w \rangle, M \text{ is a TM and } w \text{ is a string.}$

1. Simulate  $M$  on  $w$ .
2. If  $M$  accepts, accept. If  $M$  rejects, reject.

Not halting

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$



So, again, the issue here is that, how can the Turing machine or how can we ever tell whether the machine is looping? How can we tell that machine is looping? For all we know, it may be doing some computation over and over again, maybe it will stop if you let it run for another 100 steps. But how long, maybe after 100 steps it is still continuing. So, now, how long should we wait? So, when do we know whether the Turing machine is stuck on an infinite loop?

If you know that, you only have to wait for this much time, then we can wait for this much time. But otherwise, how do we know whether it is going to stick or get stuck or not? So, this seems to be the hardest part. So, telling whether  $M$  will loop on  $w$  is not that easy. Or in other words we are asking so, what is the opposite of getting stuck on a loop, the opposite of getting stuck on a loop is like being decisive and stopping or halting. So, the opposite is, can we know for sure that  $M$  halts on  $w$ . If you know  $M$  halts on  $w$  you can run it. Either, we know that it is going to halt which means it is going to accept or reject.

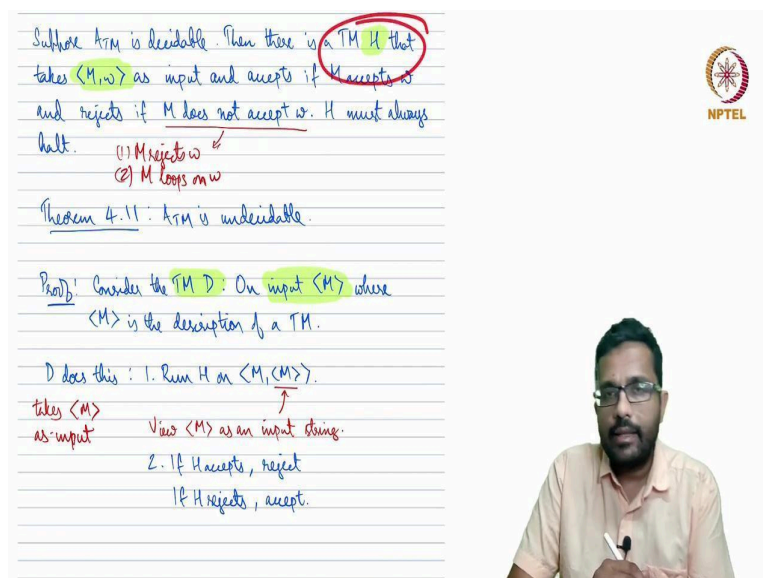
But if we know that it is not going to halt even then we know that the only way it is not going to halt is if it is looping then you can then also we can reject. So, in a sense, the real challenge is in actually telling whether or determining whether  $m$  halts on  $w$  or not. So, this is the real challenge. So, if we can tell whether  $M$  halts on  $w$ , then we just figure out whether  $M$  halts on  $w$  if it halts, then we just run the Turing machine that we described about  $U$  this one runs  $M$  on  $w$  and we know  $M$  halts on  $w$ . So, either  $M$  accepts or rejects one of these cases will happen.

Again if you know that  $M$  does not halt on  $w$  even then we know that  $M$  does not halt on  $w$  then you can immediately reject because the only way it does not halt is  $M$  loops on  $w$ . So, the challenge or the difficulty is captured by this language here which I am calling it as halt T M. So, given a Turing Machine  $M$  and a string  $w$  halt T M is asking whether  $M$  halt on  $w$ . So, given a Turing Machine  $M$  and a string  $w$  does  $M$  halt on  $w$ . So, some so, this is called the halting problem.

For our class we will also call A T M is the halting problem. And because if you can solve halt T M we can also solve A T M in the manner that they just said you decide halt T M and if  $M$  does not halt on  $w$  then you immediately reject the A T M instance because it is not going to get accepted. If you know  $M$  halts on  $w$  then we can just run  $M$  on  $w$  and let it halt either way accept or reject. So, we will so, this is the real halting problem is halting team deciding whether  $M$  halts on  $w$  but sometimes we may also refer to the A T M the language A T M as the halting problem.

So, hopefully it is clear that the difficulty in deciding whether  $a$  accepts  $w$  sorry  $M$  accepts  $w$  lies in determining whether  $M$  loops on  $w$  or not. Or in other words  $M$  halts on  $w$  or not. So, that is the challenge there. And that is why it is called the halting problem and this is also undecidable.

(Refer Slide Time: 9:14)



Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this:

1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .  
takes  $\langle M \rangle$  as input  
Views  $\langle M \rangle$  as an input string.
2. If  $H$  accepts, reject  
If  $H$  rejects, accept.

The NPTEL logo is visible in the top right corner of the slide. A video inset in the bottom right shows a man with glasses and a beard, wearing a light-colored shirt, speaking into a microphone.

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as-input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.



Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as-input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.



Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as-input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.



Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ . On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.



Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ . On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.



Halting Problem is undecidable

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \}$

One attempt:

$U =$  On input  $\langle M, w \rangle$ .  $M$  is a TM and  $w$  is a string.

- 1. Simulate  $M$  on  $w$ .
- 2. If  $M$  accepts, accept.
- If  $M$  rejects, reject.

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$ .





So, what we are going to see now is the proof that A T M is undecidable. So, we will prove it by contradiction. So, think about it. We want to show that whatever algorithm that we can try, none of these algorithms work, meaning there is no algorithm that can decide this problem. So, how do we rule out all possible algorithms? So, that is the interesting thing about this proof. So, in fact, we will use a proof that is similar to the diagonalization proof in some sense. So, when you see it, you will realise where the diagonalization happens.

So, we will assume that A T M is decidable and derive a contradiction. So, if it is decidable means there is a decider for A T M. So, let H be the decider. So, which means it takes a pair M, w as input and H accepts the pair M, w if M accepts w and rejects the pair M, w if M does not accept w. So, this has two parts, one is that M rejects w and two is M loops on w. So, in either case we want H to reject the pair M comma w. If M rejects w as well as M loops on w.

So, H is a decider which means that H never halt, sorry H never loops, H is always going to halt. So, we will derive a contradiction. So, let us see how we get the contradiction? So, the theorem is that an A T M is undecidable. So, what we do is we assume that an A T M is decidable and which means you can assume that it is a decider H like we said. Now, using that H we are going to build a machine D. So, what does D do? So, D takes some description of a Turing machine as input. So, D takes us input like the description of a Turing Machine M. That is what I have written here on input M.

So, what is this is it represents D actually runs H where H is the decider for A T M and D so, but then the input of H is in the form of a machine and a string. D gets us input a description of a machine. So, what does H get in input, H is given input. So, the machine M is the input M and the string w is also the description of M. So, we have seen that every Turing machine has a description you can encode the description of a Turing machine. So, what D does is it takes M as which is the input given to it. So, which is the M which is input for H but H also needs a string as input.

So, the string is going to be the description of the Turing Machine M. So, you have M which is a Turing machine and the description of the Turing machine is a string that is going to be fed to M.

(Refer Slide Time: 13:00)

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$



Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ . On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$  as input      Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$



Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ . On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$  as input      Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
- If  $H$  rejects, accept.

So, we run  $H$  on the Turing Machine  $M$  and the description of the Turing machine is a string. So, the same Turing Machine  $M$  appears in two ways. One is the input string and two is the sorry one is the Turing machine and two is the input of the Turing machine. So, what does  $D$  do?  $D$  runs  $H$ , which is the decider for A T M on the pair  $M$  and the description of  $m$  and then it does one more tricky thing. So,  $H$  is a decider which means we know for sure that  $H$  will accept or reject.



So, if  $M$  accepts the description of itself,  $H$  will accept if  $M$  does not accept its own description  $H$  will reject meaning, if  $M$  rejects its own description  $H$  will reject. If  $M$  loops on its own description even then  $H$  will reject. So, in either case,  $H$  will reject.

Now, what we are going to do is what does D do? D actually flips the output so, flips the output of H. So, if H accepts D rejects and if H rejects D accepts and we know for sure that H will halt. So, what does D do? D runs H on the input M and its own description as a string and if H accepts, D rejects, if H rejects, D accepts.

(Refer Slide Time: 14:34)



D does this : 1. Run H on  $\langle M, \langle M \rangle \rangle$ .  
↑  
takes  $\langle M \rangle$  as input      Views  $\langle M \rangle$  as an input string.  
2. If H accepts, reject } Flips the o/p.  
   If H rejects, accept. }

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .  
...  $\langle$  Accept if H does not accept  $\langle M \rangle$



D does this : 1. Run H on  $\langle M, \langle M \rangle \rangle$ .  
↑  
takes  $\langle M \rangle$  as input      Views  $\langle M \rangle$  as an input string.  
2. If H accepts, reject } Flips the o/p.  
   If H rejects, accept. }

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .  
...  $\langle$  Accept if H does not accept  $\langle M \rangle$



D does this : 1. Run H on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$  as input  $\uparrow$   
Views  $\langle M \rangle$  as an input string.

2. If H accepts, reject } Flips the o/p.  
If H rejects, accept. }

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .

...  $\langle$  Accept if H does not accept  $\langle M \rangle$



D does this : 1. Run H on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$  as input  $\uparrow$   
Views  $\langle M \rangle$  as an input string.

2. If H accepts, reject } Flips the o/p.  
If H rejects, accept. }

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .

...  $\langle$  Accept if H does not accept  $\langle M \rangle$



So, just to give a pictorial description, D is given M as input. So, inside D it runs the Turing Machine H which is the decider for A T M with the Turing Machine M and its own description as the input string So, then H says accept or reject and what does D do? D flips whatever H is saying and outputs. So, this is the output of D, it just flips the output of H this is what it does. So far so good.

(Refer Slide Time: 15:23)

Suppose  $A_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt.

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $A_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where  $\langle M \rangle$  is the description of a TM.

$D$  does this: 1. Run  $H$  on  $\langle M, \langle M \rangle \rangle$ .

takes  $\langle M \rangle$   
as input

Views  $\langle M \rangle$  as an input string.

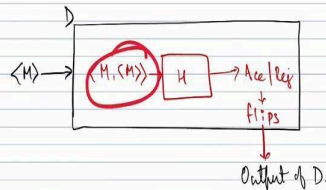
- 2. If  $H$  accepts, reject
  - If  $H$  rejects, accept.
- } Flips the o/p.



as input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
  - If  $H$  rejects, accept.
- } Flips the o/p.



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

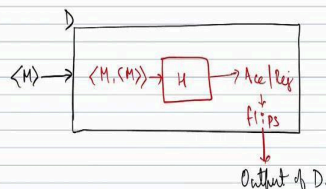
$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$$



as input

Views  $\langle M \rangle$  as an input string.

- 2. If  $H$  accepts, reject
  - If  $H$  rejects, accept.
- } Flips the o/p.

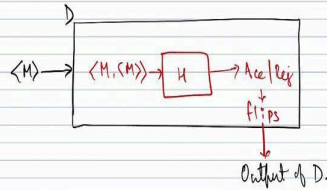


Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$$



as-input View  $\langle M \rangle$  as an input string:  
 2. If  $H$  accepts, reject } Flips the o/p.  
 If  $H$  rejects, accept.

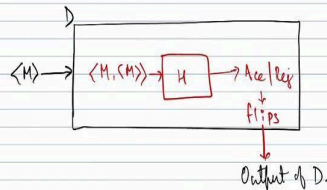


Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$

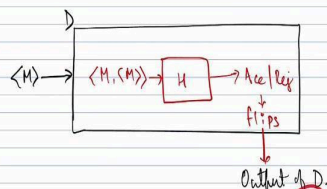


as-input View  $\langle M \rangle$  as an input string:  
 2. If  $H$  accepts, reject } Flips the o/p.  
 If  $H$  rejects, accept.



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{loop} \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$

What does  $D$  do when given  $\langle D \rangle$  as input?



So, all that we have said so far is if A T M is decidable there is a Turing Machine H which is the decider for A T M And if H is a decider for A T M using H as a building block we are making another machine D, what is D do? D takes a description of a Turing machine as input feeds H the same Turing machine and its own description as input and whatever H does it flips out. So far it is fine there is no contradiction so far. So, let us denote what let us denote by this symbol this notation  $D(\langle M \rangle)$  and parenthesis the description of M. This stands for what does d do when it is fed M as input the description of M as input.

So, let us see what D do? It takes the output of H and flips to the output. And H is a decider for A T M. So, H will accept it if M accepts its own description. So,  $D(\langle M \rangle)$  will reject if M accepts its own description and  $D(\langle M \rangle)$  will accept if M does not accept its own description So, does not accept again means two possibilities. One is reject and two is loop. So,  $D(\langle M \rangle)$  is what D do or M, it is accepted if M does not accept its own description as input. And reject if M accepts its own description as input.

So,  $D(\langle M \rangle)$  is accept if M does not accept its own description, and reject if M accepts its own description. Again, so far so far, so good. Now, where do we get the contradiction? The contradiction comes when D is given its own description as input. So, in place of M here, so we give D. So, what does D do when its own description has given us input. So, it is very simple to do with this simple to see.

(Refer Slide Time: 17:51)

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } \langle M \rangle \text{ does not accept } \langle M \rangle \\ \text{Reject if } \langle M \rangle \text{ accepts } \langle M \rangle \end{cases}$$

What does D do when given  $\langle D \rangle$  as input?



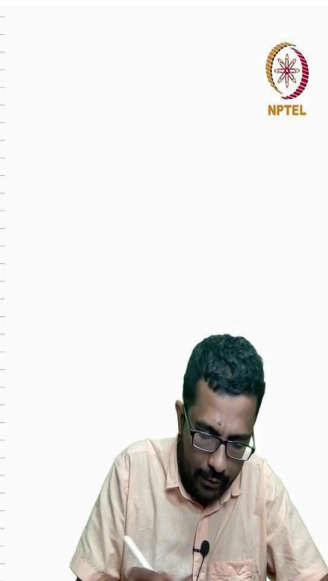
Let  $D\langle M \rangle$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D\langle M \rangle = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D\langle D \rangle = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



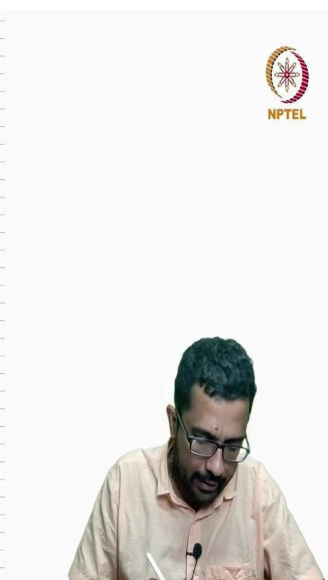
Let  $D\langle M \rangle$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D\langle M \rangle = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D\langle D \rangle = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



Let  $D\langle M \rangle$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D\langle M \rangle = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \\ \text{Reject if } M \text{ accepts } \langle M \rangle. \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D\langle D \rangle = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



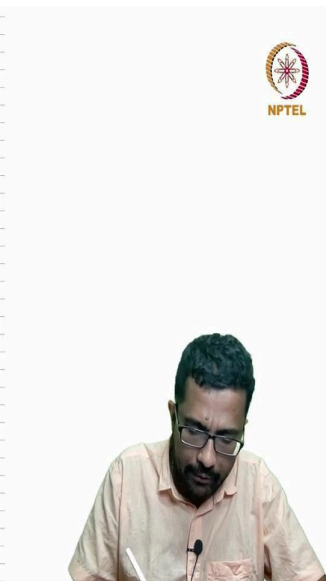
Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



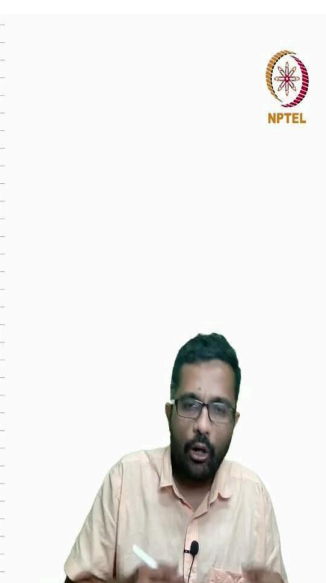
Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is



So, wherever we see M here, we just replace M with D. That is all we have to do. So, let us try doing that. So, in fact, I have already done that, I copy pasted everything that I said here, but wherever D was M was there. These 5 places I have replaced it with D, so that is also straightforward at M can be any Turing machine we replace it with D. But when you read this, now, there is an interesting thing that is going on.

So, what is the left hand side? Left hand side is what is D do when given its own description as input. This is what this means. What is D do when given its own description as input? And what do we have here? It accepts if D does not accept D and rejects D, if it if D accepts D. So, this is a very strange thing.

So, D is we are saying that D accepts its own description, D accepts D, if D does not accept D, and D rejects D, if D accepts D. So, this does not make sense. What we are saying is that whatever D does on its own description as input. D is supposed to do the opposite of that. So suppose D does not accept D, then D is supposed to accept D. Suppose D accepts D, then D is supposed to reject D. So, it does not make sense. So, that is the contradiction.

(Refer Slide Time: 19:17)

↓  
Output of D.

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .

$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$$

What does D do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So D does on  $\langle D \rangle$ , the opposite of what D is supposed to do on  $\langle D \rangle$ . This is a contradiction.

NPTEL

↓  
Output of D.

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .



$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle & \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle & \rightarrow \text{loop.} \end{cases}$$

What does D do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So D does on  $\langle D \rangle$ , the opposite of what D is supposed to do on  $\langle D \rangle$ . This is a contradiction.

U.S. ... H.S. ... A.S. ... ?

↓  
Output of D.

Let  $D(\langle M \rangle)$  denote what D does on input  $\langle M \rangle$ .



$$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle & \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle & \rightarrow \text{loop.} \end{cases}$$

What does D do when given  $\langle D \rangle$  as input?

$$D(\langle D \rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So D does on  $\langle D \rangle$ , the opposite of what D is supposed to do on  $\langle D \rangle$ . This is a contradiction.

U.S. ... H.S. ... A.S. ... ?

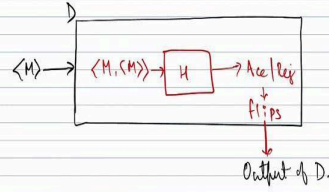



So, what we want here, what is going on here is whatever D does on its own description as input. D has to do the opposite of that. So, this just does not make sense. So, whatever D does on its own description has input it has to do its own opposite. That is a contradiction. So, maybe I will just highlight this part. D, whatever D does on its own description, description as input. D is supposed to do the opposite of that. And that is kind of an absurd statement. D does something on its own in something when it is given its own description as input. And then we are saying the deed should do the opposite. But then we will again say that it should do the same whatever it is doing now. So, that is the contradiction.

(Refer Slide Time: 20:06)

as input

2. If  $H$  accepts, reject } Flips the o/p.  
 If  $H$  rejects, accept.



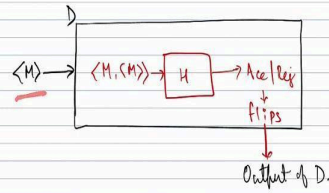
Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$



as input

2. If  $H$  accepts, reject } Flips the o/p.  
 If  $H$  rejects, accept.

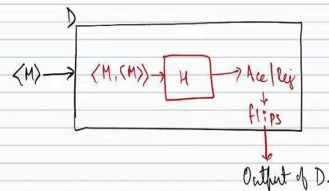


Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$



If  $H$  rejects, accept.



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .

$D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$

What does  $D$  do when given  $\langle D \rangle$  as input?



Let  $D\langle M \rangle$  denote what  $D$  does on input  $\langle M \rangle$ .

$$D\langle M \rangle = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$$

What does  $D$  do when given  $\langle D \rangle$  as input?

$$D\langle D \rangle = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D \rangle \\ \text{Reject if } D \text{ accepts } \langle D \rangle. \end{cases}$$

So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is supposed to do on  $\langle D \rangle$ . This is a contradiction.

So let me just recap because it is such an important proof.  $H$  is a decider for a T M. And what does  $D$  do, when given  $M$  as input, it feeds  $H$  with  $M$  and its own description as input, and it flips the output of it. So, what does  $D$  do when given  $M$  as input, it feeds  $M$  and its own description to  $H$ . If  $M$  accepts its own description, then  $D$  rejects it. If  $M$  does not accept its own description, then  $D$  accepts it. So,  $D$  accepts  $M$  if  $M$  does not accept its own description.  $D$  rejects  $M$  if  $M$  accepts its own description.

It is fine so far, till we decide to replace  $M$  with  $D$ . So, then  $D$  accepts its own description, if  $D$  does not accept its own description, and  $D$  rejects its own description if  $D$  accepts its own description. This is a contradiction, so this is saying that  $D$  should do the opposite of whatever it is doing right now. But when it does the opposite, we will say no, this is also not correct, because it should do the opposite of whatever it is doing right now. So, that is a contradiction.

So, this means whatever it does, it should do the opposite. And which means it is an inconsistent situation.

(Refer Slide Time: 21:30)

Why is this diagonalization?

Let  $A \in \text{TM}$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
...				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in \text{TM}$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
...				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in \text{TM}$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
...				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in TM$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in TM$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in TM$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will





Real Halting Problem  
 $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

$HALT_{TM}$  is also undecidable. We will also refer to  $HALT_{TM}$  as the "halting problem."

---

Suppose  $HALT_{TM}$  is decidable. Then there is a TM  $H$  that takes  $\langle M, w \rangle$  as input and accepts if  $M$  accepts  $w$  and rejects if  $M$  does not accept  $w$ .  $H$  must always halt:

- (1)  $M$  rejects  $w$
- (2)  $M$  loops on  $w$

Theorem 4.11:  $HALT_{TM}$  is undecidable.

Proof: Consider the TM  $D$ : On input  $\langle M \rangle$  where



1. Simulate  $M$  on  $w$ .  
 2. If  $M$  accepts, accept.  
    If  $M$  rejects, reject. Not happening

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.

$HALT_{TM}$  is Turing recognizable. However, if the TM  $H$  knows that  $M$  is going to get stuck in a loop with  $w$ , it can reject  $w$ . This is the hardest part.

Real Halting Problem  
 $HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

$HALT_{TM}$  is also undecidable. We will also refer to  $HALT_{TM}$  as the "halting problem."



So, maybe this may seem confusing. So, I suggest that maybe think over it a day or something. But I am also going to give another viewpoint for the same thing. So, if you remember we saw the explanation that real numbers are uncountable. So, we listed down the assumption that real numbers are countable, we made a listing and then by flipping the entries in the diagonal, we identified a real number, which was not captured in any of this listing. And that is the contradiction.

So, we can see this proof that A T M is decidable. So, we can see this proof as a diagonalization proof. So, notice that the contradiction so at the end, we got the contradiction, and the assumption that we are contradicting is the assumption that A T M is decidable. So, let us see another perspective for the same proof. Suppose A T M was decidable, we can

build a table. So, we have already seen that the set of all Turing machines is countable. So, now, let us build the following table.

So, the columns are descriptions of Turing machines, and the rows are indexed by Turing machines. So, we have to see the rows, the indices of the rows as Turing machines,  $M_1, M_2$  as Turing machines, and the indices for the columns as inputs. So, the inputs here are also descriptions of the Turing machines, but they can also be viewed as strings.

So, let us see how to fill this. So, what does  $M_1$  do when it is fed its own description? So, let us say it accepts what does  $M_1$  do when it does fit the description of  $M_2$ , let us say it rejects, what does  $M_1$  do when it is fed  $M_2$  is description let us say it accepts. What does  $M_1$  do when it is fed  $M_3, M_4$  description. So,  $M_1, M_2$  are all the Turing machines. So, as I already said, the set of our Turing machines is countable and we can list them.

(Refer Slide Time: 23:38)


Why is this diagonalization?


Let  $H$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will







Why is this diagonalization?

Let  $A \in TM$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will



Why is this diagonalization?

Let  $A \in TM$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$ ?

Same table as above, but all the "Loop" will

So, perhaps  $M_1$  loops on the description of  $M_4$ . So, we say loop. Similarly, what does  $M_2$  do when it is fed  $M_1$   $M_2$  do with its fed  $M_2$  and so on. So, you can fill this table with accept reject loop, accept reject loop, whatever. So, that is how we filled it. So, the reject and loop I have written in red colour, except in black colour.



(Refer Slide Time: 24:01)

Let  $A_{TM}$  be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_j \rangle \rangle$ ?

Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not

So, what does  $H$ ?  $H$  is the decider for  $A_{TM}$ . So, what does  $H$  do when there is some issue here, is given as input. What does  $H$  do when  $M_i$  and the description of  $M_j$  is given as input. So, when  $M_i$  and the description of  $M_j$  is given as input. So,  $H$  will accept if  $M_i$  accepts  $M_j$ .  $H$  will reject if  $M_i$  rejects  $M_j$ .  $H$  will also reject if  $M_i$  loops on  $M_j$ . So, it is the same table as we have here. So, what does  $M_H$  do when it is fed  $M_1$ , it will accept


What does  $H$  do when it is fed  $M_1$  comma  $M_2$  description it will reject. What does  $H$  do when it is fed  $M_1$ ,  $M_4$  this description. So, here it is loop. But then means it is not accepting  $M_1$  does not accept the description of  $M_4$ , so he will reject.

(Refer Slide Time: 25:20)

NPTEL

Arm is decidable  $\Rightarrow$  H exists  
let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle \emptyset \rangle$
$M_1$	Acc	rej	Acc	rej	...	
$M_2$	Acc	Acc	rej	rej	...	
$M_3$	rej	Acc	Acc	Acc	...	
$M_4$	rej	Acc	rej	rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
D	rej	rej	rej	Acc	...	(?)




NPTEL

accept  $\langle M_j \rangle$ , then H will reject  $\langle M_i, \langle M_j \rangle \rangle$ .

Arm is decidable  $\Rightarrow$  H exists  
let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle \emptyset \rangle$
$M_1$	Acc	rej	Acc	rej	...	
$M_2$	Acc	Acc	rej	rej	...	
$M_3$	rej	Acc	Acc	Acc	...	
$M_4$	rej	Acc	rej	rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	



Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_i \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_i \rangle \rangle$ .

$A_{TM}$  is decidable  $\Rightarrow H$  exists

let us draw the table for  $H$  on  $\langle M_i, \langle M_i \rangle \rangle$ .

$\langle M_i, \langle M_i \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	Acc	Rej	Acc	Rej	...	
$M_2$	Acc	Acc	Rej	Rej	...	
$M_3$	Rej	Acc	Acc	Acc	...	
...						

Using machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
$M_1$	Accept	Reject	Accept	Loop	
$M_2$	Accept	Accept	Reject	Loop	
$M_3$	Reject	Accept	Accept	Accept	
$M_4$	Loop	Accept	Loop	Reject	
...					

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$  is given as input?

Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_i \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_i \rangle \rangle$ .

$A_{TM}$  is decidable  $\Rightarrow H$  exists

table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
$M_1$	Accept	Reject	Accept	Loop	
$M_2$	Accept	Accept	Reject	Loop	
$M_3$	Reject	Accept	Accept	Accept	
$M_4$	Loop	Accept	Loop	Reject	
...					

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$  is given as input?

Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_i \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_i \rangle \rangle$ .



So, this table here is what does H do when it is fed  $M_i$  and the description of  $M_j$ . So, the rows indicate  $M_i$ , and the columns indicate the description of this Turing machine. So, it is the same table as before, just that whenever  $M_1$  or whenever  $M_i$  loops on  $M_j$ , we say H rejects that  $M_i$  comma  $M_j$ . Because H does not loop. H is a decider. So,  $M_i$  loops on  $M_j$  means  $M_i$  does not accept  $M_j$ . So, H rejects  $M_i$  comma  $M_j$ . So, in the same table, you see accept reject, accept loop now it is accept the first row accept reject, accept reject. Second row is accept, accept, reject, loop, over here we have accept, accept, reject, reject. So, the loop is replaced by rejects. So, we know that by assumption A T M is decidable. So, H is that.

(Refer Slide Time: 26:23)

ATM is decidable  $\Rightarrow$  H exists  
let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	rej	Acc	rej	...	
$M_2$	Acc	<u>Acc</u>	rej	rej	...	
$M_3$	rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	rej	Acc	rej	<u>rej</u>	...	
...	...	...	...	...	...	
<u>D</u>	rej	rej	rej	Acc	...	<u>?</u>



What's this entry?



ATM is decidable  $\Rightarrow$  H exists  
let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .

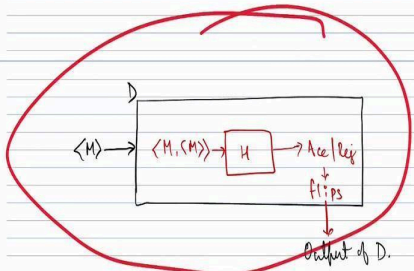
$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	<u><math>\langle D \rangle</math></u>
$M_1$	<u>Acc</u>	rej	Acc	rej	...	
$M_2$	Acc	<u>Acc</u>	rej	rej	...	
$M_3$	rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	rej	Acc	rej	<u>rej</u>	...	
...	...	...	...	...	...	
<u>D</u>	rej	rej	rej	Acc	...	<u>?</u>

What's this entry?

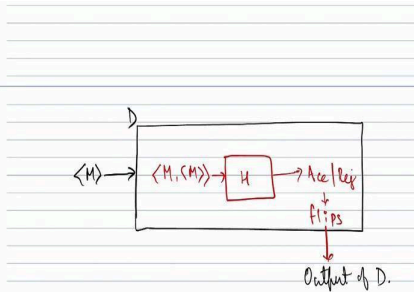




as input  
 View  $\langle M \rangle$  as an input string.  
 2. If  $H$  accepts, reject } Flips the o/p.  
 If  $H$  rejects, accept.



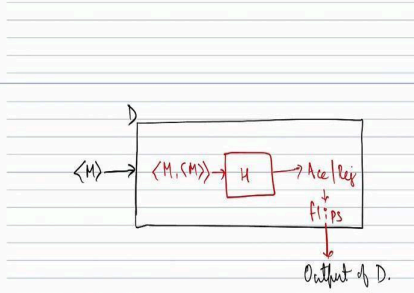
Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .  
 $D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .  
 $D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$

What does  $D$  do when given  $\langle D \rangle$  as input?

Accept if  $D$  does not accept  $\langle D \rangle$



Let  $D(\langle M \rangle)$  denote what  $D$  does on input  $\langle M \rangle$ .  
 $D(\langle M \rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M \rangle \rightarrow \text{reject} \\ \text{Reject if } M \text{ accepts } \langle M \rangle \rightarrow \text{loop.} \end{cases}$

What does  $D$  do when given  $\langle D \rangle$  as input?

Accept if  $D$  does not accept  $\langle D \rangle$



So, so far, it is just standard things. Now, where does the issue arise? So, the issue arises, when you realise that H is a decider, H exists, a decider for A T M exists. And if H exists, by the construction that we have seen over here, if H exists, I can make D from H like by building this thing around it.

So, D is just a Turing machine that is using H as a subroutine. If H exists, D also surely exists. So, which means A T M is decidable means H exists. Since H exists, D also exists, which means D is a Turing Machine, which is a decider, which means in the listing of all the Turing machines, M 1, M 2, M 3, M 4, some point D appears. So maybe it is M 100. Maybe it is M 523. We do not know.

So, the point is that in this listing D appears at some point. So, it appears in the row, it will also appear in the column and the corresponding point. So, now, how did we make D? So we wanted D to do the opposite of whatever each M i does on its own description. That is what we said here. What does D do on M, D accepts M, if M does not accept its M, and D rejects M, if M accepts M. So, we wanted D to do on each M i, the opposite of what M i itself does on its own description.

(Refer Slide Time: 28:10)

let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	Acc	rej	Acc	rej	...	
$M_2$	rej	Acc	rej	rej	...	
$M_3$	rej	Acc	Acc	Acc	...	
$M_4$	rej	rej	rej	rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
D	rej	rej	rej	Acc	...	?

What's this entry?

let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Ace</u>	rej	Ace	rej	...	
$M_2$	Ace	<u>Ace</u>	rej	rej	...	
$M_3$	rej	Ace	<u>Ace</u>	Ace	...	
$M_4$	rej	Ace	rej	<u>rej</u>	...	
...						
D	<u>rej</u>	rej	rej	Ace	...	?

What's this entry?



let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Ace</u>	rej	Ace	rej	...	
$M_2$	Ace	<u>Ace</u>	rej	rej	...	
$M_3$	rej	Ace	<u>Ace</u>	Ace	...	
$M_4$	rej	Ace	rej	<u>rej</u>	...	
...						
D	rej	<u>rej</u>	rej	Ace	...	?

What's this entry?



let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .




$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Ace</u>	rej	Ace	rej	...	
$M_2$	Ace	<u>Ace</u>	rej	rej	...	
$M_3$	rej	Ace	<u>Ace</u>	Ace	...	
$M_4$	rej	Ace	rej	<u>rej</u>	...	
...						
D	rej	rej	<u>rej</u>	Ace	...	?

What's this entry?




let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .




$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	Rej	Acc	Rej	...	
$M_2$	Acc	<u>Acc</u>	Rej	Rej	...	
$M_3$	Rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	Rej	Acc	Rej	<u>Rej</u>	...	
D	Rej	Rej	Rej	Acc	...	?

What's this entry?




let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	Rej	Acc	Rej	...	
$M_2$	Acc	<u>Acc</u>	Rej	Rej	...	
$M_3$	Rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	Rej	Acc	Rej	<u>Rej</u>	...	
D	Rej	Rej	Rej	<u>Acc</u>	...	?

What's this entry?



So, what does D do on M 1, the opposite of whatever M 1 does on M. So, M 1 accepts M 1 so, D rejects M 1. M 2 accepts M 2 so, D rejects M 2. M three accepts M 3 so, D rejects M 3. M 4 rejects M 4 so, D accepts M 4. So, it is all clear.

(Refer Slide Time: 28:29)

let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	acc	rej	acc	rej	...	
$M_2$	acc	acc	rej	rej	...	
$M_3$	rej	acc	acc	acc	...	
$M_4$	rej	acc	rej	rej	...	
...						
D	rej	rej	rej	acc	...	?

What's this entry?



let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	acc	rej	acc	rej	...	
$M_2$	acc	acc	rej	rej	...	
$M_3$	rej	acc	acc	acc	...	
$M_4$	rej	acc	rej	rej	...	
...						
D	rej	rej	rej	acc	...	?

What's this entry?



let us draw the table for H on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	acc	rej	acc	rej	...	
$M_2$	acc	acc	rej	rej	...	
$M_3$	rej	acc	acc	acc	...	
$M_4$	rej	acc	rej	rej	...	
...						
D	rej	rej	rej	acc	...	?

What's this entry?



let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle 0 \rangle$
$M_1$	Ace	Rej	Ace	Rej	...	
$M_2$	Ace	Ace	Rej	Rej	...	
$M_3$	Rej	Ace	Ace	Ace	...	
$M_4$	Rej	Ace	Rej	Rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$D$	Rej	Rej	Rej	Ace	...	?

What's this entry?

So,  $D$  is basically what  $D$  do now,  $D$  is kind of picking the diagonal entries here and  $D$  is flipping the diagonal entries. Again this is all very clear. So, because  $D$  is supposed to do the opposite of whatever. So,  $D$  what is  $D$  supposed to do on  $M$ ? The opposite of what  $M$  is supposed to do on  $M$ . So, basically  $D$  is supposed to be flipping this diagonal, which is what we have here. Again so far it is fine till you realise that  $D$  also appears as a column.

(Refer Slide Time: 29:05)

let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle 0 \rangle$
$M_1$	Ace	Rej	Ace	Rej	...	
$M_2$	Ace	Ace	Rej	Rej	...	
$M_3$	Rej	Ace	Ace	Ace	...	
$M_4$	Rej	Ace	Rej	Rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$D$	Rej	Rej	Rej	Ace	...	?

What's this entry?

let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	rej	Acc	rej	...	
$M_2$	Acc	<u>Acc</u>	rej	rej	...	
$M_3$	rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	rej	Acc	rej	<u>rej</u>	...	
...	...	...	...	...	...	
D	rej	rej	rej	Acc	...	?

What's this entry?



let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	rej	Acc	rej	...	
$M_2$	Acc	<u>Acc</u>	rej	rej	...	
$M_3$	rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	rej	Acc	rej	<u>rej</u>	...	
...	...	...	...	...	...	
D	rej	rej	rej	Acc	...	?

What's this entry?



ATM is decidable  $\Rightarrow H$  exists

let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .




3. 012  
2. 302  
5. 240...  
0. 174...

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	<u>Acc</u>	rej	Acc	rej	...	
$M_2$	Acc	<u>Acc</u>	rej	rej	...	
$M_3$	rej	Acc	<u>Acc</u>	Acc	...	
$M_4$	rej	Acc	rej	<u>rej</u>	...	
...	...	...	...	...	...	
D	rej	rej	rej	Acc	...	?

What's this entry?



$5.240\dots$   
 $0.174\dots$




$\langle M_1, \langle M_2 \rangle \rangle \rightarrow \langle M_1 \rangle \langle M_2 \rangle \langle M_3 \rangle \langle M_4 \rangle \dots \langle D \rangle$

$M_1$	Acc	Rej	Acc	Rej	...
$M_2$	Acc	Acc	Rej	Rej	...
$M_3$	Rej	Acc	Acc	Acc	...
$M_4$	Rej	Acc	Rej	Rej	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$D$	Rej	Rej	Rej	Acc	...

What's this entry? ?

$K_{TM}$  is decidable  $\Rightarrow H$  is a TM  
 $\Rightarrow D$  is a TM



So, now, the Dth row and the Dth column when you coincide now, what happens this is the interesting thing. Because, suppose this cell where Dth row and Dth column was to be accepted, but then D is supposed to flip the diagonal entry. So, the diagonal entry says accept the D. Suppose it flips it, so it should be reject. But then if it says reject then D is supposed to flip it again should we accept. So, there is nothing that we can fill here in a consistent manner. Because D by definition is supposed to flip the diagonal entry, but this itself is a diagonal entry so, it cannot flip. So, that is a contradiction.

So, now you see the parallel between the diagonalization that showed that real numbers are uncountable and this because they are also in the case of real numbers. Basically, we looked at numbers 0, 1, 2, 3 something. That is a 3.012, 2.362, 5.243 some numbers and then we picked a number let us say a which is different from 0, 6, 3 so, you picked a number 0.174 something, which differs from the first number in the first decimal place, second number, the second decimal place and so on.

Similarly, D is designed to be different from  $M_1, M_2, M_3$  and so on. But then D is now also has to be as a consequence, it also has to be different from itself. That is a contradiction. So, which means D cannot be featured in this listing. But then by assumption, we assumed that there is a decider that is the assumption that is wrong. This cell, this cell over here, we don't know what goes here.

(Refer Slide Time: 30:54)





So  $D$  does on  $\langle D \rangle$ , the opposite of what  $D$  is supposed to do on  $\langle D \rangle$ . This is a contradiction.

Why is this diagonalization?

Let ATM be decidable. Let us build the following table. As noted before, we can enumerate all the Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

So, that is a contradiction. So, maybe I will just summarise this proof once again, because it is an important profile, I do not think I think it is worth repeating. So, why is this diagonalization? So, suppose A T M is decidable, which means there is a decider for A T M. So, now, let us see we can enumerate all the Turing machines,  $M_1, M_2, M_3$  and so on, we can build a table of what each  $M_i$  does on when fed the description of  $M_j$ . So, this could be accept, reject or loop.

(Refer Slide Time: 31:36)



Turing machines  $M_1, M_2, M_3, \dots$

Machine \ Input	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle \dots$
$M_1$	Accept	Reject	Accept	Loop
$M_2$	Accept	Accept	Reject	Loop
$M_3$	Reject	Accept	Accept	Accept
$M_4$	Loop	Accept	Loop	Reject
$\vdots$				

What does  $H$  do  $\langle M_i, \langle M_i \rangle \rangle$  is given as input?

Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_i \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_i \rangle \rangle$ .

ATM is decidable  $\Rightarrow H$  exists

NPTEL

$A_{TM}$  is decidable  $\Rightarrow H$  exists  
 let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .

3.  $\emptyset 1 2$   
 2.  $3 \emptyset 2$   
 5.  $2 4 \emptyset \dots$   
 0.  $1 7 \emptyset \dots$

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_k \rangle$	$\dots$	$\langle \emptyset \rangle$
$M_1$	Acc	Rej	Acc	Rej	...	...
$M_2$	Acc	Acc	Rej	Rej	...	...
$M_3$	Rej	Acc	Acc	Acc	...	...
$M_4$	Rej	Acc	Rej	Rej	...	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\emptyset$	Rej	Rej	Rej	Acc	...	?

Now what does  $H$  do when given  $M_i$  and the description of  $M_j$  as input, what does  $H$  do when given this as input? It will be the same table except that  $H$  is decisive,  $H$  will never loop. So,  $H$  will reject when  $M_i$  loops on  $M_j$ , so it is the same table but with all loops replaced by rejects.

(Refer Slide Time: 32:01)

$M_4$     loop    accept    loop    reject  
 $\vdots$

What does  $H$  do  $\langle M_i, \langle M_j \rangle \rangle$  is given as input?



Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_j \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_j \rangle \rangle$ .

ATM is decidable  $\Rightarrow H$  exists

let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .

$3 \cdot 012$   
 $2 \cdot 302$   
 $5 \cdot 240 \dots$   
 $0 \cdot 174 \dots$

$\langle M_i, \langle M_j \rangle \rangle \rightarrow \langle M_1 \rangle \quad \langle M_2 \rangle \quad \langle M_3 \rangle \quad \langle M_4 \rangle \dots \langle 0 \rangle$



$2 \cdot 302$   
 $5 \cdot 240 \dots$   
 $0 \cdot 174 \dots$

$\langle M_i, \langle M_j \rangle \rangle \rightarrow \langle M_1 \rangle \quad \langle M_2 \rangle \quad \langle M_3 \rangle \quad \langle M_4 \rangle \dots \langle 0 \rangle$


$M_1$	<u>Acc</u>	rej	Acc	rej	...
$M_2$	Acc	<u>Acc</u>	rej	rej	...
$M_3$	rej	Acc	<u>Acc</u>	Acc	...
$M_4$	rej	Acc	rej	<u>rej</u>	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$D$	<u>Rej</u>	rej	rej	Acc	...

What's this entry?

ATM is decidable  $\Rightarrow H$  is a TM  
 $\Rightarrow D$  is a TM


2.362  
9.240...  
0.174...



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	Acc	Rej	Acc	Rej	...	
$M_2$	Acc	Acc	Rej	Rej	...	
$M_3$	Rej	Acc	Acc	Acc	...	
$M_4$	Rej	Acc	Rej	Rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$D$	Rej	Rej	Rej	Acc	...	?

What's this entry?


$ATM$  is decidable  $\Rightarrow H$  is a TM  
 $\Rightarrow D$  is a TM



Now, what by definition of  $D$ ,  $D$  is supposed to flip what  $M$  does on its own input, its own description as input. So,  $D$  is supposed to do to  $M$  the opposite of what  $M$  is supposed to do to  $M$ .  $D$  is supposed to do to  $M$  the opposite of what  $M$  is supposed to do to  $M$  so,  $D$  is supposed to accept sorry reject  $M$  1. Since  $M$  1 accepts  $M$  1.  $D$  is supposed to reject  $M$  2 because  $M$  2 accepts  $M$  2.  $D$  is supposed to reject  $M$  3 because  $M$  3 accepts  $M$  3.  $D$  is supposed to accept  $M$  4 because  $M$  4 rejects  $M$  4. The issue arises when we start worrying about what  $D$  should do on its own description.

(Refer Slide Time: 32:45)


2.362  
9.240...  
0.174...



$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
$M_1$	Acc	Rej	Acc	Rej	...	
$M_2$	Acc	Acc	Rej	Rej	...	
$M_3$	Rej	Acc	Acc	Acc	...	
$M_4$	Rej	Acc	Rej	Rej	...	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$D$	Rej	Rej	Rej	Acc	...	?

What's this entry?

$ATM$  is decidable  $\Rightarrow H$  is a TM  
 $\Rightarrow D$  is a TM



So,  $D$  is supposed to look at the diagonal entry and flip the entry but then  $D$  itself contains a diagonal entry. So, there is nothing you can fill in there in a consistent manner without a contradiction. So, that is the issue.

(Refer Slide Time: 32:58)

What does  $H$  do  $\langle M_i, \langle M_j \rangle \rangle$  is given as input?



Same table as above, but all the "Loop" will be replaced by "Reject". When  $M_i$  does not accept  $\langle M_j \rangle$ , then  $H$  will reject  $\langle M_i, \langle M_j \rangle \rangle$ .

ATM is decidable  $\Rightarrow H$  exists

Let us draw the table for  $H$  on  $\langle M_i, \langle M_j \rangle \rangle$ .

3. 012  
2. 302  
5. 240...  
0. 174...

$\langle M_i, \langle M_j \rangle \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle \emptyset \rangle$
$M_1$	Acc	Rej	Acc	Rej	...	
$M_2$	Acc	Acc	Rej	Rej	...	

So, again, this is the diagonalization based proof. It may be not that difficult, but it may be a bit tricky. So, think about it for a day or two, I think it will become clear. So, that is the proof that ATM is decidable.

(Refer Slide Time: 33:19)

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$  when  $M$  loops on  $w$  as well.



ATM is Turing recognizable. However, if the TM  $U$  knows that  $M$  is going to get stuck in a loop with  $w$ , it can reject  $w$ . This is the hardest part.


Real Halting Problem

$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$

$HALT_{TM}$  is also undecidable. We will also refer to  $HALT_{TM}$  as the "halting problem."

Suppose ATM is decidable. Then there is a TM  $H$  that



Halting Problem is undecidable

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a Turing machine and } M \text{ accepts } w \}$

One attempt:


$U = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ is a string.} \}$

1. Simulate  $M$  on  $w$ .
2. If  $M$  accepts, accept.  
If  $M$  rejects, reject.

Not halting

↑

This is only a recognizer as it will loop, when  $M$  loops on  $w$ . A decider must reject  $\langle M, w \rangle$



And in fact, as I said halt T M is also sorry A T M is undecidable. In fact, that also means that halt T M is undecidable which we can prove, but we will not do it now. And I think we are probably at the end of this lecture in this lecture, lecture number 37, where we just saw the proof that A T M is undecidable. So, there are some small things that I want to say, which I will cover in next 38. Thank you.