

Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 43
Co-Turing Recognizability

(Refer Slide Time: 0:16)

What does D do on $\langle D \rangle$? Contradiction!

Def: A is co-Turing recognizable if \bar{A} is Turing recognizable.

That is, \exists a TM M such that $L(M) = \bar{A}$. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $w \in A$. (But $L(M')$ need not be A)

Theorem 4.22: A language is decidable \Leftrightarrow

Hello and welcome to lecture number 38 of the course Theory of Computation. In lecture number 37, we saw that a TM is undecidable. So, the acceptance problem of given a Turing Machine M and the string A . w does M accept w this is not a decidable language, this is not a decidable problem. Sometimes we also refer to it as a halting problem.

(Refer Slide Time: 0:44)

If M rejects, reject.

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

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

$HALP_{TM}$ 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

And we said that the real halting problem which is given M and w does M halt on w , does M halt on w , this is also an undecidable problem. This is also an undecidable problem called T_M .

(Refer Slide Time: 1:02)

What does D do on $\langle D \rangle$? Contradiction!

Def: A is co-Turing recognizable if \bar{A} is Turing recognizable.

That is, \exists a TM M such that $L(M) = \bar{A}$. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $x \notin A$. (But $L(M')$ need not be A)

When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
 $x \notin \bar{A} \Leftrightarrow x \in A$, M can reject x
 or M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

What does D do on $\langle D \rangle$? Contradiction!

Def: A is co-Turing recognizable if \bar{A} is Turing recognizable.

That is, \exists a TM M such that $L(M) = \bar{A}$. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $x \notin A$. (But $L(M')$ need not be A)

\bar{A}	M accepts	Eqn
A	M rejects	Acc
	M loops	Loop

When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
 $x \notin \bar{A} \Leftrightarrow x \in A$, M can reject x
 or M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

In this lecture, we will just define Co-Turing recognizability and say a few remarks which will close off the fourth chapter in the book. So, let us define co-Turing recognizable. So, a language is said to be co-Turing recognizable if its complement is Turing recognizable. So, whenever this prefix co is there, it means it has something to do with its complement. So, A is co-Turing recognizable if A 's complement is Turing recognizable, which means there is a Turing Machine M that recognizes the complement of A .

Now, what does it mean? It means, so, consider the Turing Machine M that recognizes the complement of A so, when x is in A complement, or which means when x is in not in A , M accepts sorry, M accepts x . When x is not in A complement, meaning x is in A , then M , so, what we know is M recognizes A complement, so whenever x is in A complement, A must accept x , whenever x is not in A complement, M should not accept x , but then should not could be two possibilities, M is we are not saying that M should be a decider. So, M can reject x or M can loop on x , so these two possibilities exist, M can reject x or M can loop on x .

So, there must be some strings in A . So, whatever is not in A complement is in A that M may be rejecting and some strings in A that M may be looping. So, in other words, it is now if you take M and flip its output. So, all the strings that are in A complement, will be rejected. All the strings that are in A need not be accepted because there could be some strings in A that could be looping again. So, that is what I am saying here, we can flip the output of M to get a Turing Machine M prime.

So, M prime will always reject when given a string in A complement. Because whenever M is given a string in A complement, it accepts. So, whenever M complement is sorry, M prime is given a string in A complement, it will certainly reject, but it may not accept every string in A because M may not need not reject everything in A and may loop on some strings in A .

So, in other words, what I am saying may be very quickly, you can draw a figure. So, there are three types of strings. So, this is A complement, this is A . So, in A complement all the strings M accepts. In A there are strings that M rejects. And there are strings that M loops. So, these two together form A . These two sets together form A .

Now the point is that when we flip the output, this accept will become reject. And this reject will become accept, but the loop will remain loop. So, that is why I am saying that for M prime, it is not. The language recognized need not be A , it is some subset of A . But all the things in A complement will certainly be rejected. Anyway, co-Turing recognizable means that the complement must be Turing recognizable.

So, one of the interesting theorems is that a language is decidable. If and only if the language is Turing recognizable, and is also co-Turing recognizable. So, A is decidable if and only if it is Turing recognizable, and co-Turing recognizable. So, let us see the proof on why this is true.

(Refer Slide Time: 5:55)

\bar{A} & A . (But $L(M')$ need not be \bar{A})

\bar{A}	M accepts x	Eqn
A	M rejects x	Acc
	M loops	loop



When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
 $x \notin A \Leftrightarrow x \in \bar{A}$, M can reject x
OR M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow
it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M decider
with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get
 M' , a decider for \bar{A} .

$\Rightarrow \bar{A}$ is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.



\bar{A} & A . (But $L(M')$ need not be \bar{A})

\bar{A}	M accepts x	Eqn
A	M rejects x	Acc
	M loops	loop



When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
 $x \notin A \Leftrightarrow x \in \bar{A}$, M can reject x
OR M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow
it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider
with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get
 M' , a decider for \bar{A} .

$\Rightarrow \bar{A}$ is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.



Q & A. (But $L(M')$ need not be A)



\bar{A}	Accepts	Exp	When $x \in A \Leftrightarrow x \notin A$, M accepts x $x \notin A \Leftrightarrow x \in A$, M can reject x OR M can loop on x .
A	Rejects	Acc	
	M loops	loops	

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get M' , a decider for \bar{A} .

$\Rightarrow \bar{A}$ is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.

Q & A. (But $L(M')$ need not be A)



\bar{A}	Accepts	Exp	When $x \in A \Leftrightarrow x \notin A$, M accepts x $x \notin A \Leftrightarrow x \in A$, M can reject x OR M can loop on x .
A	Rejects	Acc	
	M loops	loops	

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get M' , a decider for \bar{A} .

$\Rightarrow \bar{A}$ is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.

So, this is an if and if and only if statement, so we need to prove both directions. So, first, we will prove the forward direction. Suppose the language is decidable. We will show that it is both Turing recognizable and co-Turing recognizable. So, suppose it is decidable. Suppose A is decidable, which means there is a decider for A let us say the decider is M , which means the language recognized by M is A . So, M is also a recognizer for A . That is it. Now, we have to show that A is co-Turing recognizable also.

M is a decider for A so, M always halts which means M accepts all the strings in A and rejects all the strings not in A . So, we can take M and flip the output of M . So, it will always reject strings that are in A and accepts strings that are not in A . So, in fact, if you flip the output of M , you will get M' which is a decider for A complement. So, a decider for A complement will also be a recognizer for A complement. Hence, A complement is Turing

recognizable, which means is co-Turing recognizable. So, the decider for A if you flip the output, you get a decider for A complement, which is a recognizer for A complement, which means A is co-Turing recognizable.

So, the forward direction is complete. If A is decidable, the decider for A sorry the recognizer for A . And if you flip the output, you get a recognizer for A complement. Now, the other direction is, if a language is both recognizable and co-Turing recognizable, we have to show that it is decidable.

(Refer Slide Time: 7:40)



A is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.

Suppose A is both Turing recognizable and co-Turing recognizable.

$\exists M$ such that $L(M) = A$
 $\exists M'$ such that $L(M') = \bar{A}$

Now we will see how to build a decider for A .
Given input w , run both M and M' in parallel (one step each). Given any w , it is either in A or \bar{A} . If it is in A , M accepts w . If it is in \bar{A} , M' accepts w . So for any w , either M or M' must halt.

If M accepts, accept.
If M' accepts, reject.





A is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.

Suppose A is both Turing recognizable and co-Turing recognizable.

$\exists M$ such that $L(M) = A$
 $\exists M'$ such that $L(M') = \bar{A}$

Now we will see how to build a decider for A .
Given input w , run both M and M' in parallel (one step each). Given any w , it is either in A or \bar{A} . If it is in A , M accepts w . If it is in \bar{A} , M' accepts w . So for any w , either M or M' must halt.

If M accepts, accept.
If M' accepts, reject.





Suppose A is both Turing recognizable and co-Turing recognizable.

$\exists \text{ TM } M \text{ such that } L(M) = A$
 $\exists \text{ TM } M' \text{ such that } L(M') = \bar{A}$

Now we will see how to build a decider for A .
 Given input w , run both M and M' in parallel (one step each). Given any w , it is either in A or \bar{A} . If it is in A , M accepts w . If it is in \bar{A} , M' accepts w . So for any w , either M or M' must halt.

If M accepts, accept. M
 If M' accepts, reject. M'

Given w , run M and M' in parallel. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $w \notin A$. (But $L(M')$ need not be \bar{A})



\bar{A}	Accepts	Loop
A	Rejects	Accepts
	M loops	M loops

When $x \in \bar{A} \Leftrightarrow x \notin A$, M accepts x
 $x \notin \bar{A} \Leftrightarrow x \in A$, M can reject x
 OR M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get M' , a decider for \bar{A} .

So, suppose, A is both recognizable and co-Turing recognizable. So, what does it mean? It means that there is a recognizer for A , there is a recognizer for A complement, they need not be related machines, there could be two different machines. So, let us say M is the recognizer for A , which means, M is a Turing machine that accepts all the strings in A and does not accept any string that is not in A . Again, it may reject or loop on the strings not in A because it is not it may not be a decider.

Similarly, let M prime be the decider for A complement, it accepts all the strings in A complement, it may reject or loop on the strings that are not in A complement, or which in other words that are in A . So, now using these two recognizes, recognizer for A , recognizer for A complement, we can build a decider for A . So, the decider is very simple. What we will do is we will run M and we will run M prime both in parallel. So, on the same input, we will

not make the mistake of running one completely and waiting for the M instead will run in parallel, we will run one step M, one step M prime and so on. So, it is like both of them progress in computation.

The point is this given A string, it is either in A or in A complement, because A and A complement, A complement is just defined to be everything that is not A, so, given any string is either in A or in A complement. Suppose it is in A. So, given a string w suppose it is in A which means we know that M accepts it because M is a recognizer for A, so we know that M must accept w. If it is not in A which means it is in A complement, we know that M prime accepts that because M prime is a recognizer for A complement. So, if w is not in A, w must be accepted by M prime.

So, given any string, either it has to be accepted by M or it has to be accepted by M prime. So, we run M and M prime in parallel, at some point one of them must accept. So, if M accepts, we accept, it is a string in. So, we want a decider for A. So, we want all the strings in A to be accepted, and all the strings not in A to be rejected.

So, and we know that any string that is given as input is either going to be accepted by M or going to be accepted by M prime. So, if it is accepted by M, we accept, because it is a string in A. If it is accepted by M prime, it has to be a string in A complement that we reject. And one of this must happen because both every string is either in A or A complement, if it is in A, M should accept if it is in A complement, M prime should accept.

So, running these two machines in parallel, one of them must accept and if depending on which one you accept, which one is accepting, we this we say it is an A or A complement. So, this is a decider, because if it is an A it is accepted, if it is not an A it is rejected. So, we are combining a recognizer for A and a recognizer for A complement to build a decider for A.

So, again, just stating the result, the result is again that a language is decidable if and only if it is both Turing recognizable, and co-Turing recognizable, so, decidable if these two are satisfied then it has to be decidable.

(Refer Slide Time: 11:43)

M' accepts w . So for any w , either M or M' must halt.

If M accepts, accept.

If M' accepts, reject.

M

M'



Cor 4.23: \bar{A}_{TM} is not Turing recognizable.

Proof: A_{TM} is undecidable.

A_{TM} is Turing recognizable.

If \bar{A}_{TM} was Turing recognizable, then by Theorem 4.22 above, A_{TM} will be decidable. Hence \bar{A}_{TM} is not Turing recognizable.



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. *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.



Def: A is co-Turing recognizable if \bar{A} is Turing recognizable.

That is, \exists a TM M such that $L(M) = \bar{A}$. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $w \in A$. (But $L(M')$ need not be A)

\bar{A}	Accepts x	When $x \in \bar{A} \Leftrightarrow x \notin A$, M accepts x
A	Rejects x	$x \notin \bar{A} \Leftrightarrow x \in A$, M can reject x
	M loops	OR M can loop on x .

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

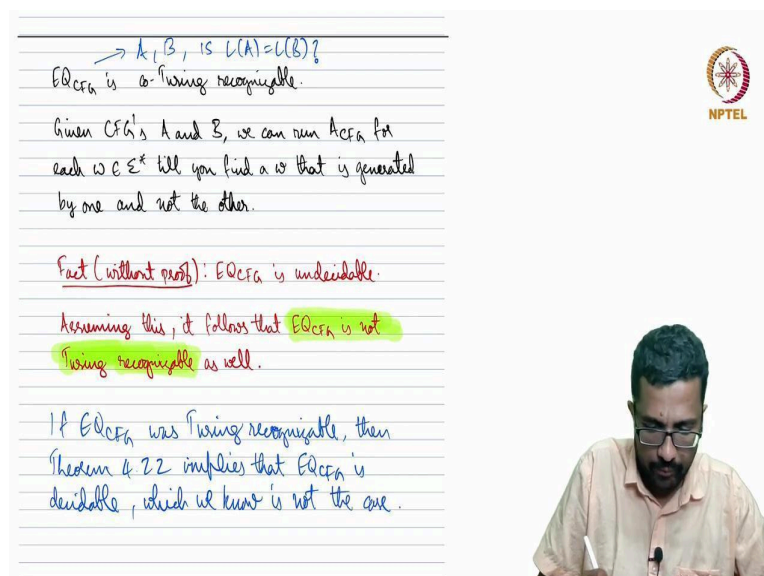


So, finally, some minor observations using this theorem. So, we saw at the beginning of lecture 37 that A T M is recognizable. So, this Turing machine that we talked about U this is a recognizer for A T M, because whenever M accepts w, it is a yes instance of A T M, U accept that pair. The issue happens when M loops on w. So, A T M is Turing recognizable. We have shown that A T M is undecidable. So, if A T M is co-Turing recognizable as well, or in other words, if A T M complement was also Turing recognizable, both A T M and A T M complement will be Turing recognizable, this would imply that A T M is decidable. But we know that A T M is not decidable.

So, we cannot have both an A T M and its complement being recognizable. Let me repeat. So, the theorem that we just saw said that a language is decidable if it is both recognizable and co-Turing recognizable. So, we know that A T M is recognizable. If A T M complement is also recognizable, that would imply that A T M is decidable. We know that an A T M is not decidable. So, that should mean that necessarily A T M is not co-Turing recognizable which means A T M complement is not Turing recognizable, because if A T M complement was co-Turing recognizable, it will imply that A T M is decidable. Hence, we can infer that A T M complement is not Turing recognizable.

So, this is the first time we are seeing a specific language and saying that this is not Turing recognizable. A T M was decidable but it is sorry A T M was undecidable, but it was recognizable. So, here is a language the complement of A T M which is not even recognizable.

(Refer Slide Time: 13:52)




→ A, B, is $L(A) = L(B)$?
 EQ_{CFG} is a Turing recognizable.


Given CFG's A and B, we can run A_{CFG} for each $w \in \Sigma^*$ till you find a w that is generated by one and not the other.

Fact (without proof): EQ_{CFG} is undecidable.

Assuming this, it follows that EQ_{CFG} is not Turing recognizable as well.

If EQ_{CFG} was Turing recognizable, then Theorem 4.22 implies that EQ_{CFG} is decidable, which we know is not the case.





Given CFG's A and B , we can run A for each $w \in \Sigma^*$ till you find a w that is generated by one and not the other.

Fact (without proof): EQ_{CFG} is undecidable.

Assuming this, it follows that EQ_{CFG} is not Turing recognizable as well.

If EQ_{CFG} was Turing recognizable, then Theorem 4.22 implies that EQ_{CFG} is decidable, which we know is not the case.

Hence EQ_{CFG} is not Turing recognizable.

Finally, one couple of more points. We asked if or we mentioned that EQ_{CFG} is not decidable. And we are saying that EQ_{CFG} is co-Turing recognizable meaning it will be the complement of EQ_{CFG} is recognizable. So, why is it recognizable? So, if you just to remind ourselves EQ_{CFG} is like you are given A and B context free grammars. So, given A and B is L_A equal to L_B . Are they equivalent? This is the question. So, if they are not equal and so I am talking about co-Turing recognizable, it is a no instance.

Suppose it is not equivalent, we can just run through all possible strings. And there has to be some string that is accepted by A or generated by A but not generated by B or vice versa generated by B not generated by A . So, you just run through all these things starting from empty string then string of length 1 string of length 2 and so on. At some point you will find a string that is accepted by one but not by the other. If they are not equivalent, you will find such a string. So, that is why the complement of EQ_{CFG} the no instance when A and B are not equivalent, is Turing recognizable.

In other words, EQ_{CFG} is Co-turing recognizable. So, the complement of EQ_{CFG} is recognizable, which means EQ_{CFG} co-Turing recognizable by this process. And I mentioned earlier that EQ_{CFG} is undecidable. Right now we are not going to prove it. But right now let us take it as a fact.

Now EQ_{CFG} is undecidable. But it is also co-Turing recognizable. If it was co-Turing recognizable that would imply that because it is Turing recognizable and co-Turing recognizable, it is decidable but we know that it is not decidable. Hence, it implies that EQ_{CFG} is not Turing recognizable. Maybe I will explain again if EQ_{CFG} was Turing recognizable then

theorem 4.22 implies that EQ_{CFG} is decidable which we know is not the case. Hence EQ_{CFG} is not Turing recognizable. So, that is the last point that I want to say.

(Refer Slide Time: 17:35)



Def: A is co-Turing recognizable if \bar{A} is Turing recognizable.

That is, \exists a TM M such that $L(M) = \bar{A}$. We can flip the output of M to get a TM M' , such that M' always halts and rejects when $x \notin A$. (But $L(M')$ need not be A)

\bar{A}	M accepts	Eg	When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
	M rejects	Acc	
A	M loops	Loop	

Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider

never halts \Rightarrow always accepts and rejects when $x \notin A$. (But $L(M')$ need not be A)

\bar{A}	M accepts	Eg	When $x \in A \Leftrightarrow x \notin \bar{A}$, M accepts x
	M rejects	Acc	
A	M loops	Loop	



Theorem 4.22: A language is decidable \Leftrightarrow it is Turing recognizable and co-Turing recognizable.

Proof: A is decidable $\Rightarrow \exists$ a TM M , decider
with $L(M) = A$
 $\Rightarrow A$ is Turing recognizable.

M always halts. So flip the output of M to get M' , a decider for \bar{A} .

$\Rightarrow \bar{A}$ is Turing recognizable.

$\Rightarrow A$ is co-Turing recognizable.

So, just to recap what we saw in lecture 38, we define co-Turing recognizable, A is co-Turing recognizable, if A complement a Turing recognizable, then we show this theorem that a language is decidable if it is both recognizable and co-Turing recognizable.

(Refer Slide Time: 17:48)

Cor 4.23: $\overline{A_{TM}}$ is not Turing recognizable.

Proof: A_{TM} is undecidable.
 A_{TM} is Turing recognizable.

If $\overline{A_{TM}}$ was Turing recognizable, then by Theorem 4.22 above, A_{TM} will be decidable. Hence $\overline{A_{TM}}$ is not Turing recognizable. (A_{TM} is not co-Turing recognizable)

→ A, B , is $L(A) = L(B)$?
 EQ_{CFG} is co-Turing recognizable.

Given CFG's A and B , we can run A and B for each $w \in \Sigma^*$ till you find a w that is generated by one and not the other.

NPTEL

And using this, we inferred that we saw the proof. Using this we inferred that A_{TM} complement is not Turing recognizable, or in other words, A_{TM} is not co-Turing recognizable, so maybe I will just write that here. A_{TM} is not co-Turing recognizable.

(Refer Slide Time: 18:16)

is not Turing recognizable. (A_{TM} is not co-Turing recognizable)

→ A, B , is $L(A) = L(B)$?
 EQ_{CFG} is co-Turing recognizable.

Given CFG's A and B , we can run A and B for each $w \in \Sigma^*$ till you find a w that is generated by one and not the other.

Fact (without proof): EQ_{CFG} is undecidable.

Assuming this, it follows that EQ_{CFG} is not Turing recognizable as well.

If EQ_{CFG} was Turing recognizable, then Theorem 4.22 implies that EQ_{CFG} is decidable, which we know is not the case.

NPTEL

And similarly we saw that EQ_{CFG} is not Turing recognizable, because it is co-Turing recognizable and undecidable.

(Refer Slide Time: 18:24)

\neg is Turing recognizable.
 $\Rightarrow A$ is co-Turing recognizable.

Suppose A is both Turing recognizable and co-Turing recognizable.

\exists TM M such that $L(M) = A$
 \exists TM M' such that $L(M') = \bar{A}$

Now we will see how to build a decider for A .
Given input w , run both M and M' in parallel (one step each). Given any w , it is either in A or \bar{A} . If it is in A , M accepts w . If it is in \bar{A} , M' accepts w . So for any w , either M or M' must halt.

If M accepts, accept. M
If M' accepts, reject. M'

And that is where we end lecture 38. This also the end of week seven's content. There is also the end of chapter 4 in the book. So, in chapter 4, and in week 7 so, week 7 was this entirely chapter 4, we saw decidable languages using regular languages, decidable languages using context free languages. We saw the theory of countable and uncountable sets, we saw undecidable languages, we saw that A T M is undecidable. We saw the proof for that. And now we are seeing some other results about co-Turing recognizable Turing recognizable as well. So, this completes chapter 4, and also week 7.

In chapter 5, which we will see in week 8, we will see reductions, which is a way to transform one problem to another. And because one of them is easy, the other problem is easy. And because one of them is hard, the other problem is hard. We can make relative, we can make inferences about the relative difficulty of these problems using these transformations.

So, at some point, we had seen some aspects of these transformations earlier. At that point I had indicated, these are called reductions. So, we will see that in chapter 5 and as far as week 8, as far as the week 7 is concerned, this is it. So, see you next week.