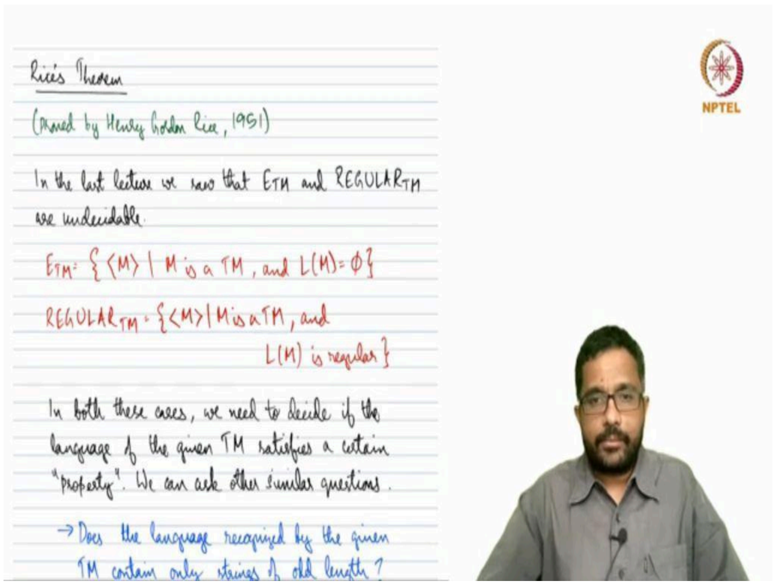
Theory of Computation

Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Rice Theorem

(Refer Slide Time: 00:16)



Rice's Theorem

(Proved by Henry Gordon Rice, 1951)

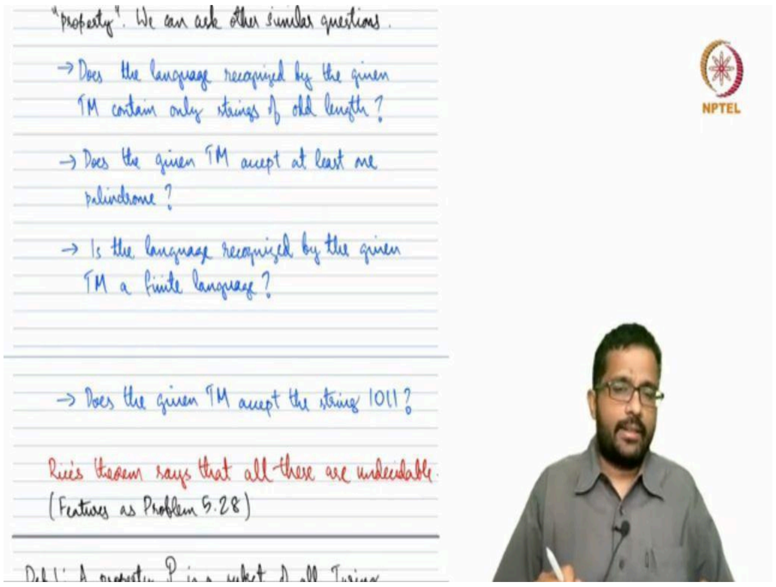
In the last lecture we saw that E_{TM} and $REGULAR_{TM}$ are undecidable.

$E_{TM} = \{\langle M\rangle \mid M$ is a TM, and $L(M) = \phi\}$

$REGULAR_{TM} = \{\langle M\rangle \mid M$ is a TM, and $L(M)$ is regular$\}$

In both these cases, we need to decide if the language of the given TM satisfies a certain "property". We can ask other similar questions.

→ Does the language recognized by the given TM contain only strings of odd length?



"property". We can ask other similar questions.

→ Does the language recognized by the given TM contain only strings of odd length?

→ Does the given TM accept at least one palindrome?

→ Is the language recognized by the given TM a finite language?

→ Does the given TM accept the string 1011?

Rice's theorem says that all these are undecidable.
(Features as Problem 5.28)

Defn 1: A property P is a subset of all Turing

Hello and welcome to lecture 41 of the course Theory of Computation. In the previous lecture, we saw many undecidable languages, and from reductions using reductions from A_{TM}. In particular, we saw two undecidable languages E_{TM} and REGULAR $_{TM}$. So, E_{TM} is the question. So, given a Turing machine, is the language recognized by this Turing machine the empty language? That is what the team is asking. Given a Turing machine, does this Turing

Machine recognize the empty language?? And REGULAR $_{TM}$ is given a Turing machine is the language recognized by this Turing machine regular.

So, these are the two languages that we saw last time. And we saw that both of them were undecidable. So, one way to look at both these problems or both these questions is like you are given a Turing machine, and you are asked whether the language recognized by this Turing machine satisfies some property. So, in one case we are asking if the language is equal to the empty language, in the other case, we are asking if this language belongs to the class of regular languages. So, we can ask many such questions of similar type. Given a Turing machine, does the language recognized by this Turing machine contain only strings of odd length? So, that is a question.

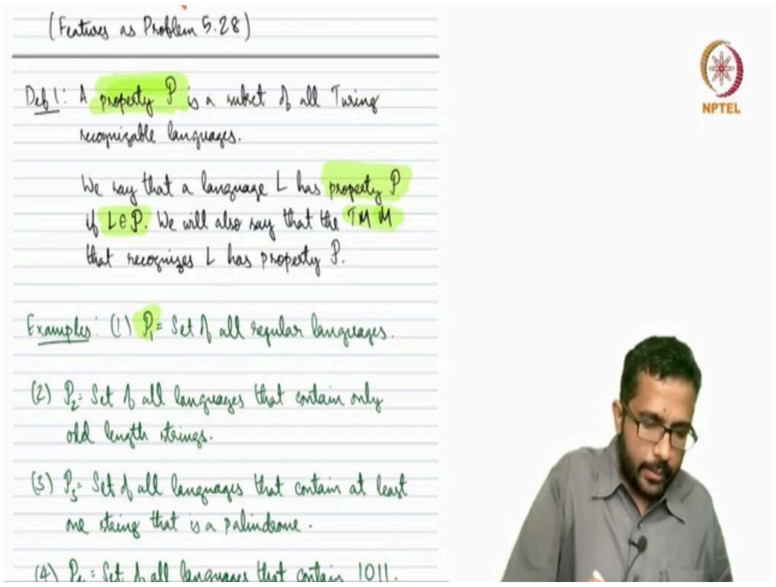
So, so this is a question that is specific to only the language of the Turing machine. So, I am not asking you about other things like how many states it have, or how the transitions happen? Or is there a state where it loops or anything of that sort? The questions that I am talking about are only specific to the language. So once again, the question that I am talking about is does the given a Turing machine, does the language recognizable the Turing machine contain only strings of odd length? Another similar question is given a Turing machine does a given Turing machine accept at least one Palindrome string?

So, palindrome is a string that reads the same way from left to right or from right to left. Another question that I can ask is, given a Turing machine is the language recognized by this Turing machine? Is it a finite language? Does it only accept a finite number of strings? So, that is another question. Or, does the given Turing machine accept a specific string, let us say 1011? So, this is another question. Because so again, we are asking whether 1011 belongs to the language recognized by this Turing machine? So, all these questions are questions specific to the language recognized by the Turing machine.

And so, we saw two questions in the previous lecture and we saw that both of them are undecidable, E_{TM} and REGULAR $_{TM}$. What Rice's theorem says is that all these questions that I have listed here, all of them are undecidable. So, this is actually not covered in the textbook per se, but it appears as a problem, and I think the solution is then the book itself. But, it is not given in as much detail as any regular content of the text, because it just gives a solution not much of explanation, So, the and the Rice's theorem states that all these are undecidable.

So, given a Turing machine, if you want to ask, does this, does the language recognized by this Turing machine does it satisfy a certain property? The answer is going to be almost always this is undecidable. So, given a Turing machine, it is undecidable to determine whether the language recognized by this Turing machine satisfies a property. And I say almost always and not always because there are some minor cases where there is an exception, so, these are going to be very easy to figure out. So, let me just now formally go to the statement of the theorem. So, hopefully this gives you an idea of the informal understanding of what the theorem states. So, given a Turing machine we are asking, does the language recognized by this Turing machine satisfy a certain property, and to determine this, answer to this is undecidable.

(Refer Slide Time: 04:54)



(Features as Problem 5.28)

Def 1: A property P is a subset of all Turing recognizable languages.

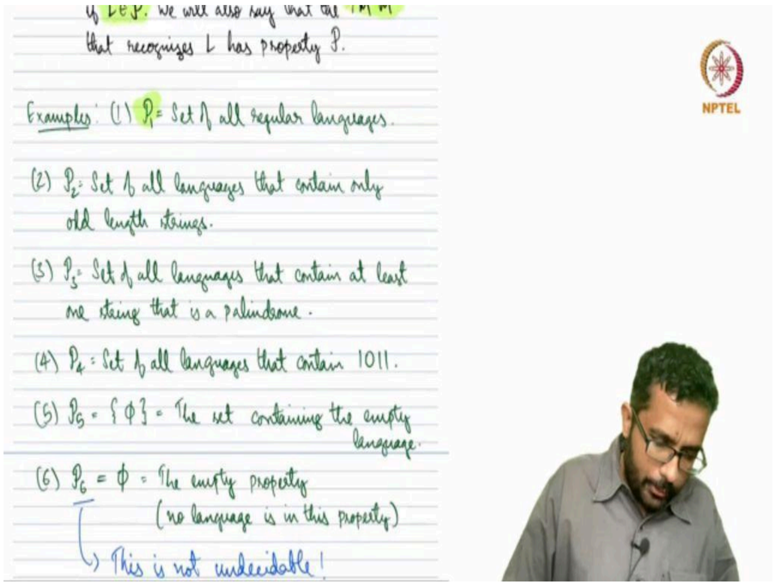
We say that a language L has property P if $L \in P$. We will also say that the TM M that recognizes L has property P.

Examples: (1) P_1 = Set of all regular languages.

(2) P_2 = Set of all languages that contain only odd length strings.

(3) P_3 = Set of all languages that contain at least one string that is a palindrome.

(4) P_4 = Set of all languages that contain 1011.



if $L \in P$. We will also say that the TM M
that recognizes L has property P.

Examples: (1) P_1 = Set of all regular languages.

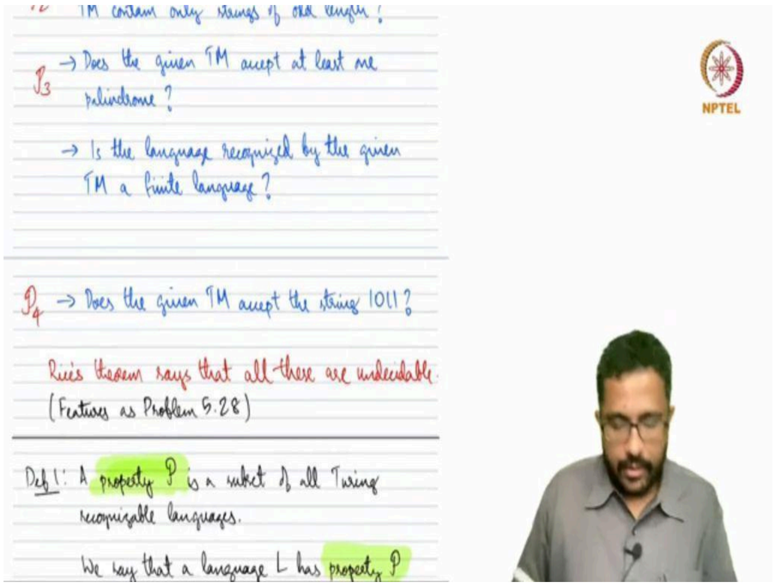
(2) P_2 = Set of all languages that contain only odd length strings.

(3) P_3 = Set of all languages that contain at least one string that is a palindrome.

(4) P_4 = Set of all languages that contain 1011.

(5) $P_5 = \{\phi\}$ = The set containing the empty language.

(6) $P_6 = \phi$ = The empty property
(no language is in this property)
$\hookrightarrow$ This is not undecidable!



TM contain only strings of odd length?

P_3 → Does the given TM accept at least one palindrome?

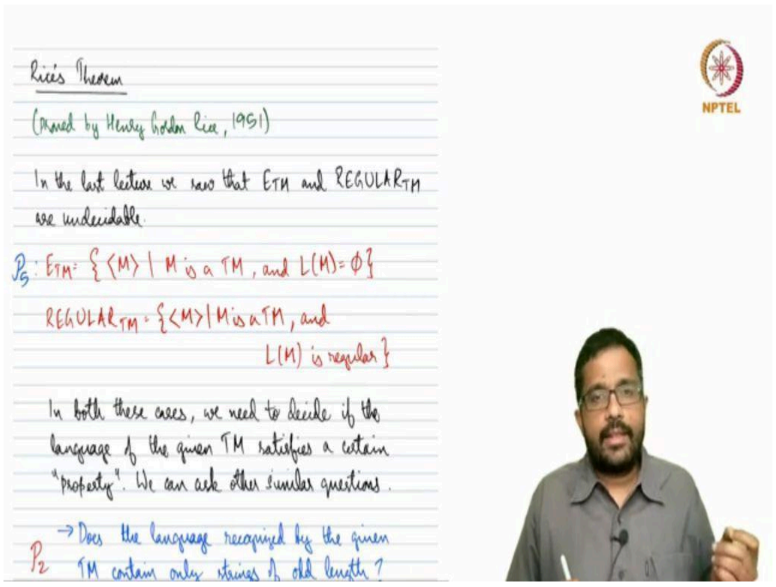
→ Is the language recognized by the given TM a finite language?

P_4 → Does the given TM accept the string 1011?

Rice's theorem says that all these are undecidable.
(Features as Problem 5.28)

Def 1: A property P is a subset of all Turing recognizable languages.

We say that a language L has property P


Rice's Theorem

(Proved by Henry Gordon Rice, 1951)

In the last lecture we saw that E_{TM} and $REGULAR_{TM}$ are undecidable.

P_5: $E_{TM} = \{\langle M\rangle \mid M$ is a TM, and $L(M) = \phi\}$

$REGULAR_{TM} = \{\langle M\rangle \mid M$ is a TM, and $L(M)$ is regular$\}$

In both these cases, we need to decide if the language of the given TM satisfies a certain "property". We can ask other similar questions.

P_2 → Does the language recognized by the given TM contain only strings of odd length?

So, what is it what do I mean by a property? A property is nothing but a subset of Turing recognizable languages. So, for instance, being regular is a property. So, not all languages like only some languages are regular, so that is a property, being finite is a property. A language is finite if it has finitely many elements. And we say that a language L has property P, so property P is a subset of languages if L belongs to that property. We also say that the Turing machine itself has that property. So, we say that M the Turing Machine M also has that property. So, instead of saying the language is recognized by M has that property, sometimes we may say that M has that property.

The language recognized by M has this property, so, give some examples. So, one I already said the set of regular languages is one property, and we saw in the previous lectures that to ask whether the Turing machine satisfying this property is undecidable. Another property is set of all languages whose members are only odd length strings, so set of all languages whose members are only odd length strings. So, this is the same as this question, so, this question is asking the first question that is listed here. It is asking whether the given Turing machine satisfies property P2, P2 means P2 is a set of all languages that contain only odd length strings.

The question here is, does the Turing machine, does the language recognized by the Turing machine contain only strings of odd length strings. P3 is set of all languages that contain at least one string that is a palindrome. And p3 is the same question asked here. Does a given Turing Machine accept at least one palindrome? So maybe I will just note it down here. So, this is asking, does a given Turing machine satisfy P2? This is asking whether the given Turing machine satisfies P3, and P4 is a set of all languages that contain 1011. So, this question is asking whether the Turing Machine satisfies P4, so, that is what we are asking. So, all these questions whether the Turing machine satisfies this property, or whether the language recognized by the Turing machines satisfies this property, are all undecidable.

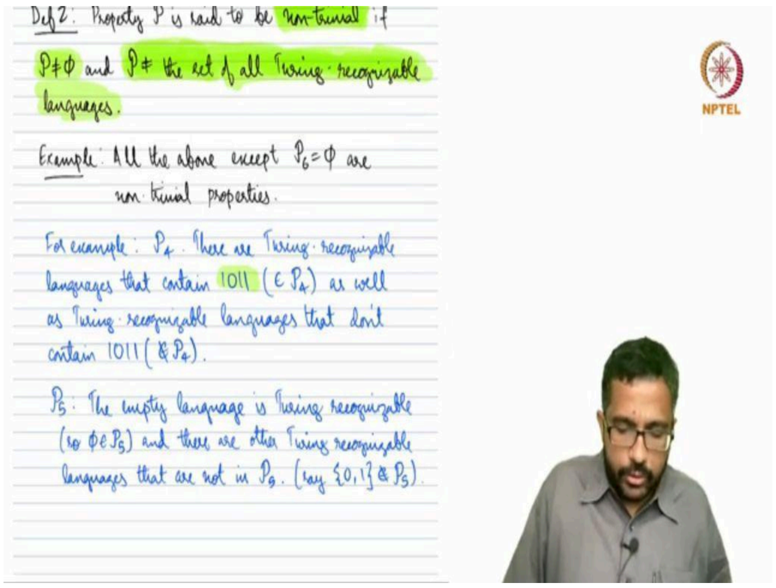
That is Rice's theorem. Now, what is P5? P5 is a set containing the empty language. So, P5 is only one language is there in P5, it is the empty language, so it is the same. So, here we are asking we will use a different colour just to stand out. Does the given Turing machine recognize the empty language? So then, if it recognizes the empty language, it satisfies P5, because P5 contains only the empty language. So, notice that here there is an empty set contained in a set. So, the set containing only the empty set, where this empty set denotes the empty language. So, this is the set of which languages satisfy this property, the only empty, only the empty language satisfies this property.

P6 is the empty set. So, notice the distinction between this and this P5 is a set containing the empty language. P6 is the empty set, meaning it is the property itself is empty, meaning no language satisfies this property. So, no language is contained in P6 that is P6, so however, this is not going to be undecidable.

This is not undecidable. When I say this, I mean, given a Turing machine it is not undecidable whether to check whether the language recognized satisfies P6. Because we know that no language satisfies P6. So, any Turing machine we can say does not satisfy P6, because whatever be the language it recognizes, it is not going to satisfy P6.

So, given a Turing machine you just check that it is a Turing machine, and you can say it does not satisfy P6, so it is not undecidable. Anyway, so, once again I want to highlight the difference between this and this. So, P5 is the property that contains only one language which is the empty language. P6 is the property is the empty property itself, meaning no language satisfies this property.

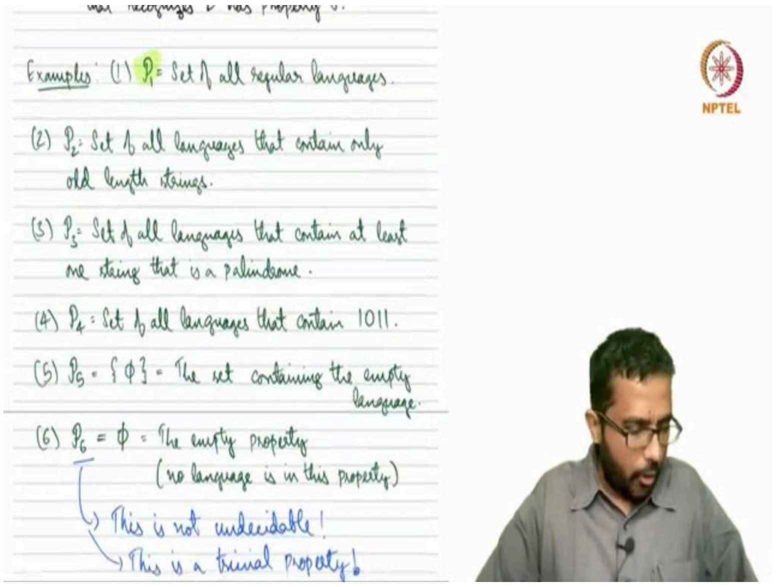
(Refer Slide Time: 10:11)


Def 2: Property P is said to be non-trivial if $P \neq \phi$ and $P \neq$ the set of all Turing-recognizable languages.

Example: All the above except $P_6 = \phi$ are non-trivial properties.

For example: P_4. There are Turing-recognizable languages that contain 1011 ($\in P_4$) as well as Turing-recognizable languages that don't contain 1011 ($\notin P_4$).

P_5. The empty language is Turing recognizable (so $\phi \in P_5$) and there are other Turing recognizable languages that are not in P_5. (say $\{0,1\} \notin P_5$).


that recognizes L has property P.

Examples: (1) P_1 = Set of all regular languages.

(2) P_2: Set of all languages that contain only odd length strings.

(3) P_3: Set of all languages that contain at least one string that is a palindrome.

(4) P_4: Set of all languages that contain 1011.

(5) $P_5 = \{\phi\}$ = The set containing the empty language.

(6) $P_6 = \phi$ = The empty property (no language is in this property)

→ This is not undecidable!

→ This is a trivial property!

So, we say that a property is non-trivial. If there is at least one language that satisfies that property, meaning that property is not empty. And to the property is not all the Turing machine , all the Turing recognizable languages, meaning so, two extremes. One is a (prop), if the property is empty, so P6 was the empty property.

So, that is the kind of trivial case, no language satisfies this, so we can say no Turing machine also satisfies this. It is a trivial thing, so we do not even need to spend effort to check. Second is the set of all Turing recognizable languages. So, given any Turing machine, it will recognize the language that it recognizes, they will be part of the set of all Turing recognizable languages, so that is also trivial.

So, between these two extremes, these two extremes, what is the empty property where no language satisfies this? And two is the set of all Turing recognizable languages satisfying this. Between these two extremes, any other property that so the two things, one is that it must contain at least one language, so it is not empty.

Two is that it should not contain all the Turing recognizable languages. So, there should be at least one in the language and one out of the language, one Turing recognizable language that satisfies this property and one Turing recognizable language that does not satisfy this property. If these two things are there, then it is not, then it is said to be non-trivial.

So, I do not want the property to be all the Turing recognizable languages, or I do not want the property to be nothing also. So, otherwise, there is no, there is nothing interesting there. So, we want not everything, not a thing. We want it to contain some language and it to not contain some language also. So, it could be just one language that it is not containing, or one language is containing, that is. So, all this all the properties that we listed here, except P6 are non-trivial. So, P6 is a trivial property because P6 is equal to empty. So P4, for instance, P4 is a set of all languages that contain 1011.

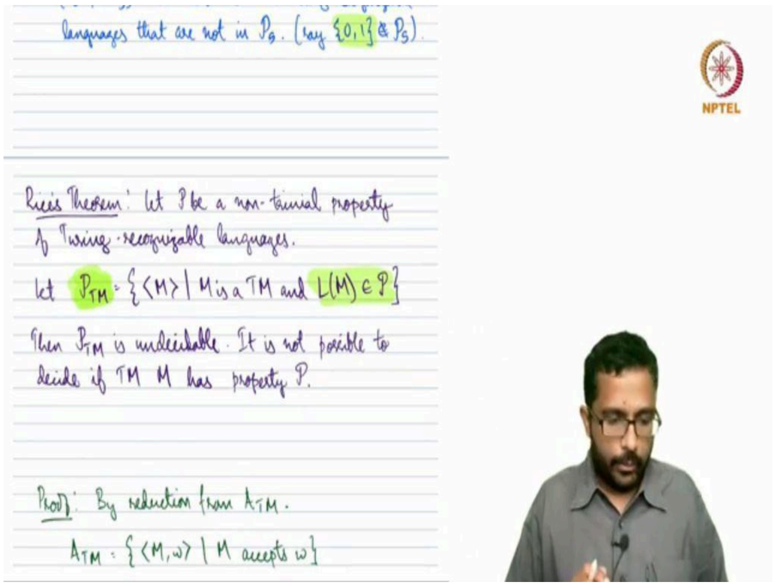
So, I can make a set of languages that are Turing recognizable, that contain 1011, you can easily make something. You can also make Turing recognizable languages that do not contain 1011. So, you can have some other language, it is such a set of all palindromes, it does not contain 1011, but it is Turing recognizable.

And set of all numbers, set of all strings that contain an odd number of 1s that contains 1011. So, that is a Turing recognizable language that satisfies P4. So, P4 is non-trivial. There is a Turing recognizable language that contains 1011, and there is a Turing recognizable language that does not contain 1011.

Another one is P5, what is P5? P5 is a set containing only the empty language. So, empty language is there in P5, and any other language if you take it is not there in P5. So, the set of all palindromes is not there in P5, so set of all, because P5 only contains empty language. Or, set of the language is containing two strings ‘0’ ‘1’, just two strings that is also not there in P5.

So, P5 is also non-trivial. So, all the properties listed here are not trivial except P6. So, this is not a non-trivial property, rather, it is a trivial property, so, trivial properties are easy to check. Given a Turing machine, if it is an empty language, it is you can say it is certain, empty property you can say, the language does not satisfy this. All Rice's theorem says is that given a non-trivial property, it could be any one of them, or it could be there are many more that you can think of P1 to P5, they are all non-trivial properties.

(Refer Slide Time: 14:43)



languages that are not in P_5. (say $\{0,1\} \notin P_5$).

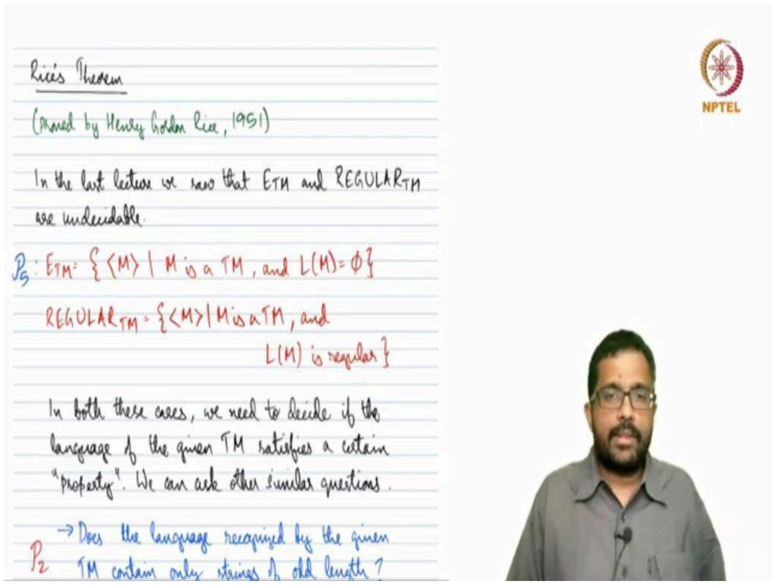
Rice's Theorem: Let P be a non-trivial property of Turing-recognizable languages.

Let $P_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \in P\}$

Then P_{TM} is undecidable. It is not possible to decide if TM M has property P.

Proof: By reduction from A_{TM}.

$A_{TM} = \{\langle M, w \rangle \mid M \text{ accepts } w\}$



Rice's Theorem

(Proved by Henry Gordon Rice, 1951)

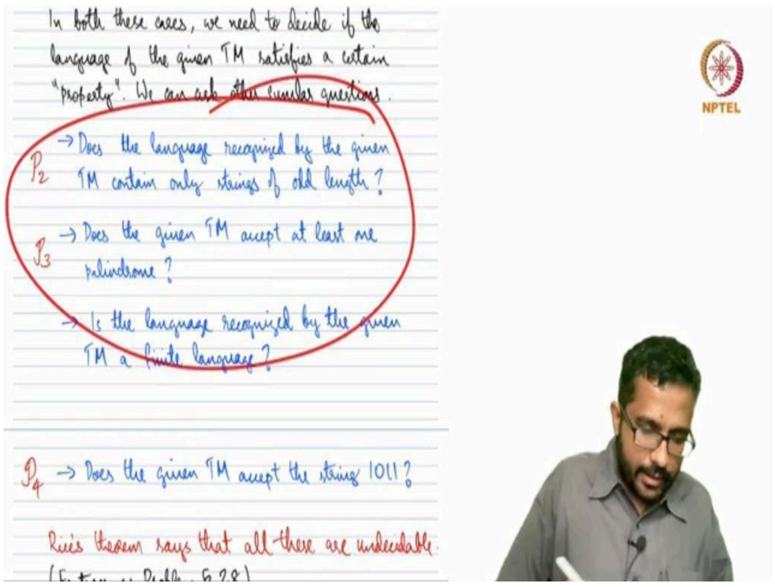
In the last lecture we saw that E_{TM} and $REGULAR_{TM}$ are undecidable.

P_5: $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM, and } L(M) = \phi\}$

$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM, and } L(M) \text{ is regular}\}$

In both these cases, we need to decide if the language of the given TM satisfies a certain "property". We can ask other similar questions.

P_2 → Does the language recognized by the given TM contain only strings of odd length?


In both these cases, we need to decide if the language of the given TM satisfies a certain "property". We can ask other similar questions.

P_2 → Does the language recognized by the given TM contain only strings of odd length?

P_3 → Does the given TM accept at least one palindrome?

→ Is the language recognized by the given TM a finite language?

P_4 → Does the given TM accept the string 1011?

Rice's theorem says that all these are undecidable.

[illegible]

Now, consider the question PTM. So, what is PTM? Given a Turing machine M, does M satisfy the property or does the language recognized by M satisfy the property is that in P? Then, PTM is undecidable. Given a Turing machine, it is undecidable to check whether the Turing machine satisfies this property, or the language recognized by the Turing machine satisfies this property.

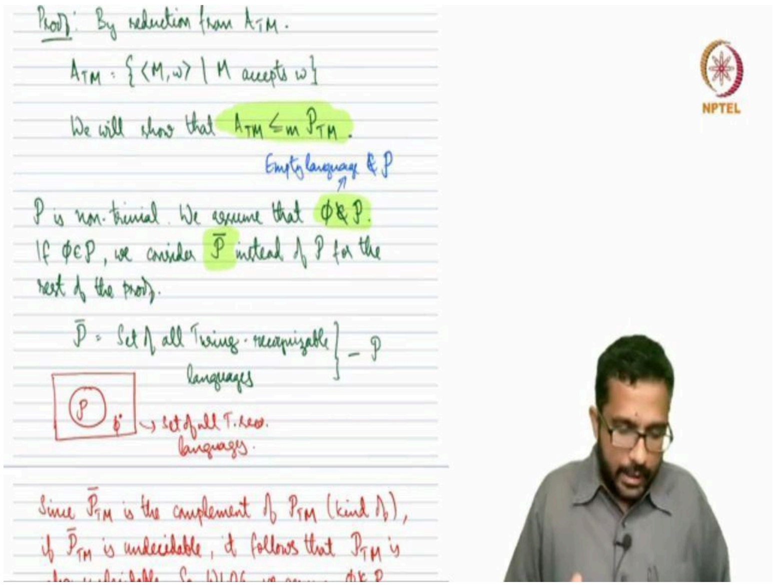
For any non-trivial property that is Rice's theorem. So, automatically, in the last class whatever we did like E_{TM} is undecidable, REGULAR $_{TM}$ is undecidable. Both of them now immediately follow as consequences of Rice's theorem, because being empty is a property and being regular is another property, they are both non-trivial properties.

So, immediately by application of Rice's theorem, we know that E_{TM} is undecidable and REGULAR $_{TM}$ is undecidable. And now by whatever I have said, all of this, all of these questions that are asked here, does the language contain only strings of odd length? Does the language contain at least one palindrome? Does the language recognized by the Turing machine contain 1011? All of these are undecidable questions. So, Rice's theorem says that it is undecidable to check whether a given Turing Machine satisfies a non-trivial property. So, that is the significance and that is the kind of explanation of what the statement of Rice's theorem is.

So, it is as you can think of it is very very powerful, because not only did we prove that two already proven undecidable languages are undecidable. Now, I am saying a very huge set of languages, anything of this type, any question that you can think of. Now, given a Turing machine is the language recognized by this Turing machine is it context free? Given a Turing machine, does it contain any string that ends in 000? Does it recognize any string that does it accept anything that ends in 000? Given a Turing machine does it recognize two strings x and y, such that x is the complement of y?

So, I can make any number of questions and all of these are undecidable, because they are all non-trivial properties. So, given that now, let us move to the proof of Rice's theorem. So, the statement I have explained so far. The proof is also not that complicated, it is brief, it is interesting.

(Refer Slide Time: 17:29)



Proof: By reduction from A_{TM}.

$A_{TM} = \{\langle M,w\rangle \mid M \text{ accepts } w\}$

We will show that $A_{TM} \leq_m P_{TM}$.

Empty language $\notin P$

P is non-trivial. We assume that $\phi \notin P$.
If $\phi \in P$, we consider $\overline{P}$ instead of P for the rest of the proof.

$\overline{P} = \left.\begin{matrix}\text{Set of all Turing-recognizable} \\ \text{languages}\end{matrix}\right\} - P$

$\hookrightarrow$ set of all T.recog. languages.

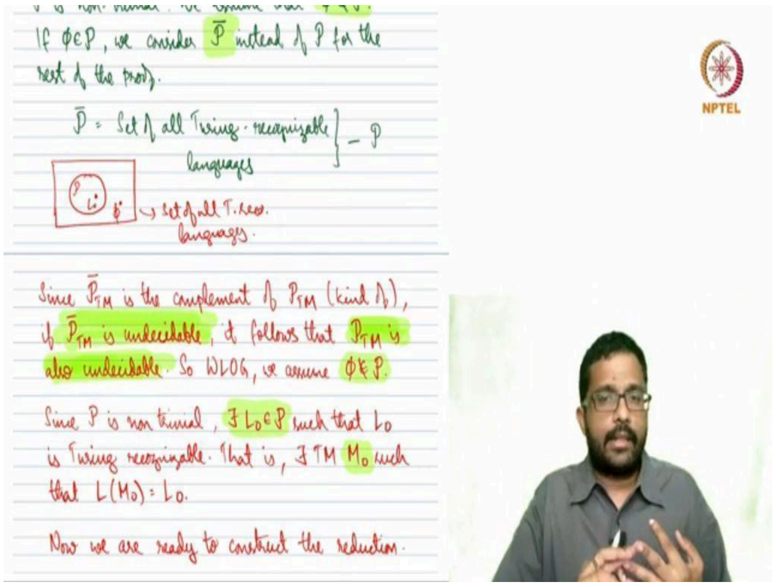
Since $\overline{P}_{TM}$ is the complement of P_{TM} (kind of), if $\overline{P}_{TM}$ is undecidable, it follows that P_{TM} is

It is by reduction from A_{TM}, so, A_{TM} we have already seen. Given <M,w>, we want to know whether M accepts w. So, we will show that A_{TM} reduces to P_{TM}, P_{TM} is the property TM. So, and this by results earlier shown implies that P_{TM} is undecidable because A_{TM} is undecidable. So, by the statement of the theorem, P is a non-trivial property, which means it is not the empty property and it is not the property of all the languages, all the Turing recognizable languages. So, we can let us first assume that the empty language is not in P. So, maybe I will just write here, empty language is not in P, so, it is so, this is a without loss of generality assumption.

So, suppose empty language was in the property. So, now what we are going to do is we can take the complement of this property. So, when I say a complement, I am talking about a P complement. So, when I say complement what I mean is, you take the set of all Turing recognizable languages minus this(P).

So, suppose this is P, suppose this set of all Turing recognizable languages. So, now, if an empty set is here, sorry empty language is here, then we can consider P complement. Meaning now, so now asking whether a Turing machine has a property P and asking whether it does not, asking whether it has the property P complement is the same. Because, if it has the property P, it is not have the property P complement, if it has the property P complement, it does not have the property P. So, to decide P is the same as deciding P complement.

(Refer Slide Time: 19:48)



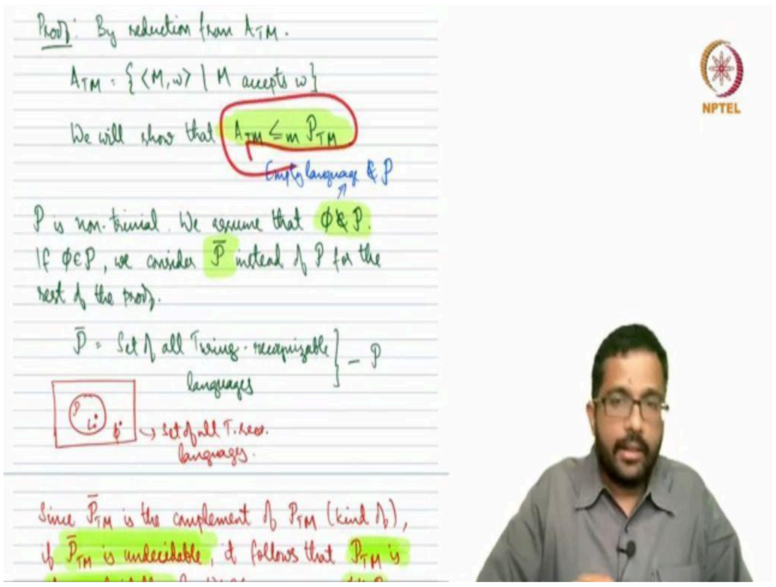
If $\phi \in P$, we consider $\overline{P}$ instead of P for the rest of the proof.

$\overline{P}$ = {Set of all Turing-recognizable languages} − P

Since $\overline{P}_{TM}$ is the complement of P_{TM} (kind of), if $\overline{P}_{TM}$ is undecidable, it follows that P_{TM} is also undecidable. So WLOG, we assume $\phi \notin P$.

Since P is non trivial, $\exists L_0 \in P$ such that L_0 is Turing recognizable. That is, $\exists$ TM M_0 such that $L(M_0) = L_0$.

Now we are ready to construct the reduction.



Proof: By reduction from A_{TM}.

$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

We will show that $A_{TM} \leq_m P_{TM}$

Empty language $\notin P$

P is non-trivial. We assume that $\phi \notin P$.

If $\phi \in P$, we consider $\overline{P}$ instead of P for the rest of the proof.

$\overline{P}$ = {Set of all Turing-recognizable languages} − P

Since $\overline{P}_{TM}$ is the complement of P_{TM} (kind of), if $\overline{P}_{TM}$ is undecidable, it follows that P_{TM} is

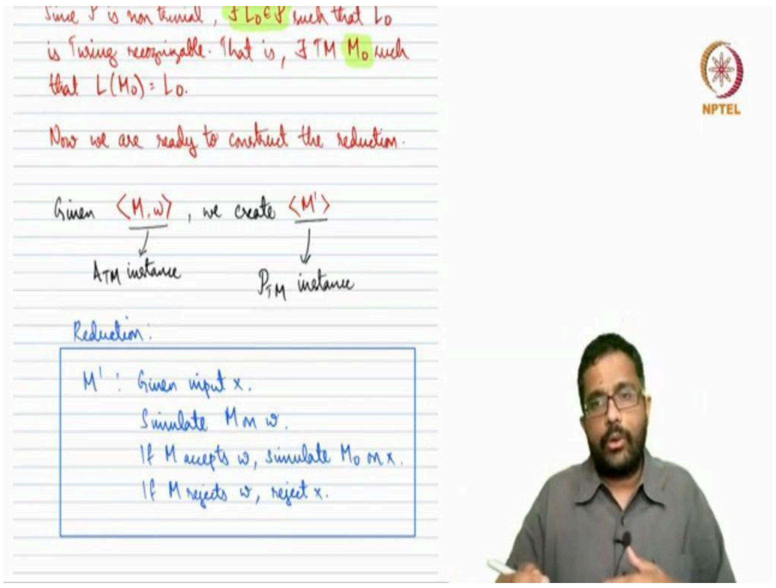
So, P_{TM}, so what I am saying is that we can consider P complement instead of P. So, if P does not contain the empty language, we can consider complement, and P complement certainly contains the empty language and then proceed. And then deciding P complement is the same as deciding P. So, it cannot be that P_{TM} is decidable and P complement TM is undecidable, because they are like you just flip yes and no, the answer is the same. If you flip yes or no, the answer for P_{TM} becomes the answer for P complement. So, if P_{TM} is undecidable, it follows that P_{TM} is also undecidable.

So, it is enough to show that P complement TM is undecidable. So, now, because of this thing, we can without loss of generality, so, again WLOG stands for without loss of generality. Without loss of generality we can assume that empty language is not in the property. So, basically what we are doing is we know that P is not the set of all Turing recognizable languages, we want basically we want something outside the language, so that we are going to be the empty language. We also want something in the language, so that we are going to pick to be some language called L0.

So, L0 is a language that is going to be in P, so, L0 is going to be language that is in P, and empty set, empty language is a language that is not in P. So, now this these two languages the empty language and the L0 is going to be important in our proof. So, we can assume that there is some language in the property, because it is not the empty property.

Because empty properties are trivial, so, let that be L0, L0 is in P. And since it is L0, it is a Turing recognizable language, let M0 be the machine Turing machine that recognizes L0. So, we have the empty language not in P and we have the language L0 in P. And L0 is recognized by some Turing machine called M0. So now, we will make the reduction, the reduction is from A_{TM} to P_{TM}, from A_{TM} to P_{TM}.

(Refer Slide Time: 22:30)



Since $\mathcal{P}$ is non trivial, $\exists L_0 \in \mathcal{P}$ such that L_0 is Turing recognizable. That is, $\exists$ TM M_0 such that $L(M_0) = L_0$.

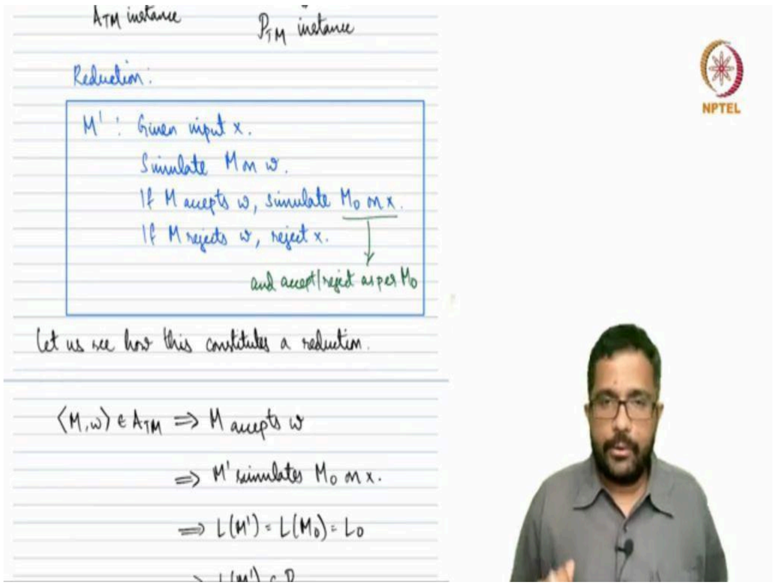
Now we are ready to construct the reduction.

Given $\langle M, w\rangle$, we create $\langle M'\rangle$

$$\begin{tikzcd} \langle M,w\rangle \arrow[d] & \langle M'\rangle \arrow[d] \\ A_{TM} \text{ instance} & P_{TM} \text{ instance} \end{tikzcd}$$

Reduction:

```
M' : Given input x.
     Simulate M on w.
     If M accepts w, simulate M_0 on x.
     If M rejects w, reject x.
```



A_{TM} instance P_{TM} instance

Reduction:

```
M' : Given input x.
     Simulate M on w.
     If M accepts w, simulate M_0 on x.
                     ↓ and accept/reject as per M_0
     If M rejects w, reject x.
```

Let us see how this constitutes a reduction.

$\langle M, w\rangle \in A_{TM} \Rightarrow M$ accepts w

$\Rightarrow M'$ simulates M_0 on x.

$\Rightarrow L(M') = L(M_0) = L_0$

$\Rightarrow L(M') \in \mathcal{P}$

So, given an A_{TM} instance, meaning given M comma w, we will construct a P_{TM} instance. So, the P_{TM} instance that we are going to be constructing is called M', this is M'. So, given an A_{TM} instance, we are going to construct a P_{TM} instance. So, let us see how to build the P_{TM} instance from the A_{TM} instance. So, the reduction is very brief, not that much, it is very brief. It is what is written in the blue box here. So, what does M' do? So, whenever it is given an input x, it runs or it simulates M on w. So, where M' basically M' has the A_{TM} instance encoded in it.

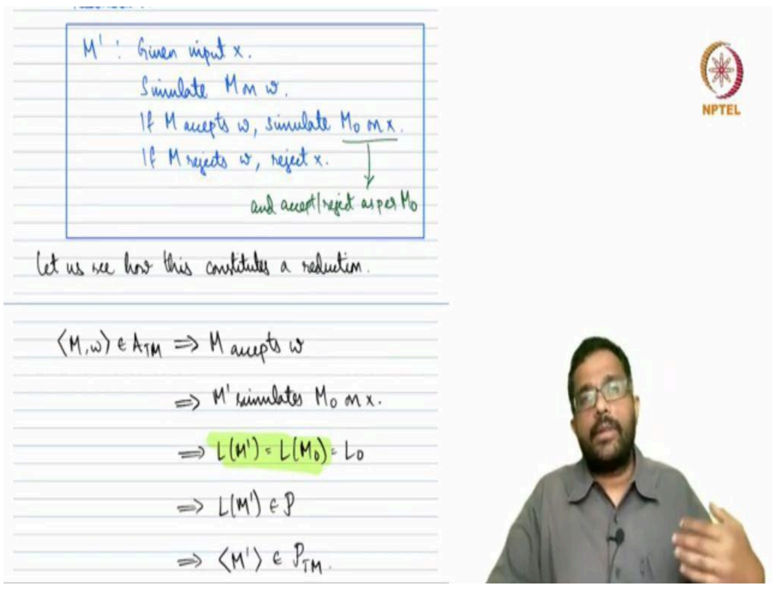
A_{TM} instance M and w encoded in it, sorry, M and w encoded in it. So, it runs M on w. So, x is the input to M', M' runs M on w, so, not on X but on w. Now, M may accept, reject or loop on w. If M accepts w, then M' is starts simulating M0 on the input given to it, where M0 is this M0, it is a Turing machine that recognizes L0. So, if M accepts w, then M' runs M0 on x.

So, now is when the input of M' is starting to get operated on, it runs M0 on x. If M' rejects w, now M' sorry if M rejects w, M' rejects x. And if M loops on w, then it just then it does not have to do anything, it just continues looping.

So, given the input M' simulates M and w, if M accepts, M' further goes ahead and simulates M0 on x. And once this happens when it simulates M0 on x, it will do whatever M0 on x, M0 does on x, it does. So, if M0 accepts, it accepts M' accepts, so, maybe I will just note that here. And basically, whatever and thus, and accept/reject as per M0. If M0 accepts x, M, M' accepts x, if M0 rejects x, M, M' rejects x. Now, if M that is if M accepts w, if M rejects W, then M' rejects x. So, let us see why this is a reduction, so, now it is just a bunch of rules, but let us see why this is a valid reduction.

So, to understand this we need to understand what happens. It is not that difficult, we just need to think about what happens when M accepts w and what happens when M does not accept w. These are the yes no cases of M, M comma W being in ATM, and not being in ATM. So, let us see in the first case what happens when M accepts w.

(Refer Slide Time: 26:27)



M' : Given input x.
Simulate M on w.
If M accepts w, simulate M_0 on x.
If M rejects w, reject x.
and accept/reject as per M_0

Let us see how this constitutes a reduction.

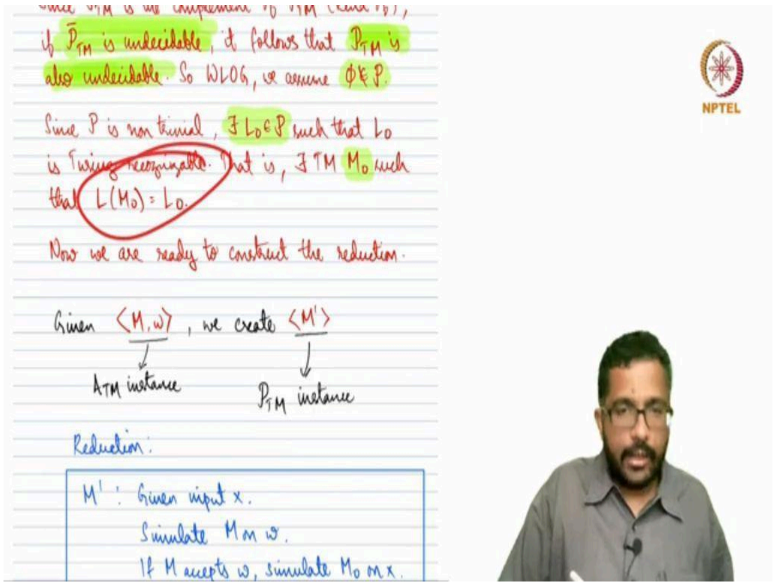
$\langle M,w\rangle \in A_{TM} \Rightarrow$ M accepts w

$\Rightarrow$ M' simulates M_0 on x.

$\Rightarrow L(M') = L(M_0) = L_0$

$\Rightarrow L(M') \in P$

$\Rightarrow \langle M'\rangle \in P_{TM}$.


if $\overline{P_{TM}}$ is undecidable, it follows that P_{TM} is also undecidable. So WLOG, we assume $\phi \notin P$.

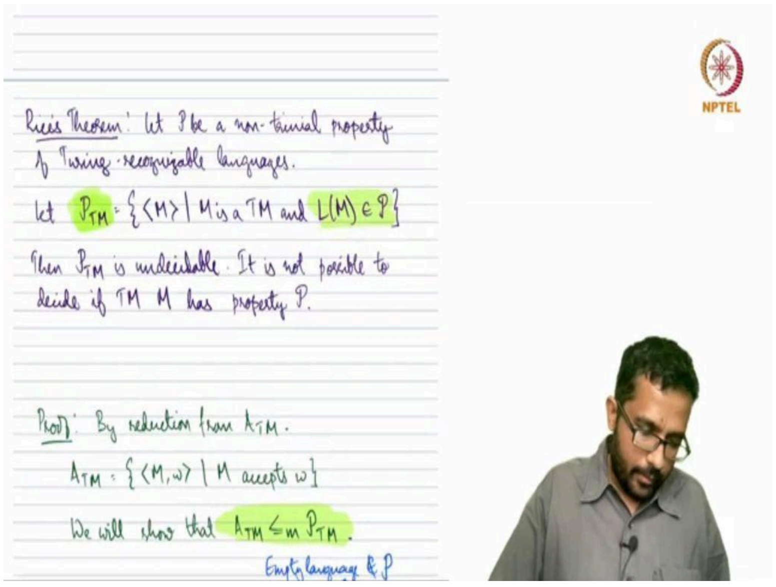
Since P is non trivial, $\exists L_0 \in P$ such that L_0 is Turing recognizable. That is, $\exists$ TM M_0 such that $L(M_0) = L_0$.

Now we are ready to construct the reduction.

$$\begin{tikzcd} \text{Given } \langle M,w\rangle \arrow[d] & \text{, we create } \langle M'\rangle \arrow[d] \\ A_{TM} \text{ instance} & P_{TM} \text{ instance} \end{tikzcd}$$

Reduction:

M' : Given input x.
Simulate M on w.
If M accepts w, simulate M_0 on x.


Rice's Theorem: Let P be a non-trivial property of Turing-recognizable languages.

Let $P_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \in P\}$

Then P_{TM} is undecidable. It is not possible to decide if TM M has property P.

Proof: By reduction from A_{TM}.

$A_{TM} = \{\langle M,w\rangle \mid M \text{ accepts } w\}$

We will show that $A_{TM} \le_m P_{TM}$.

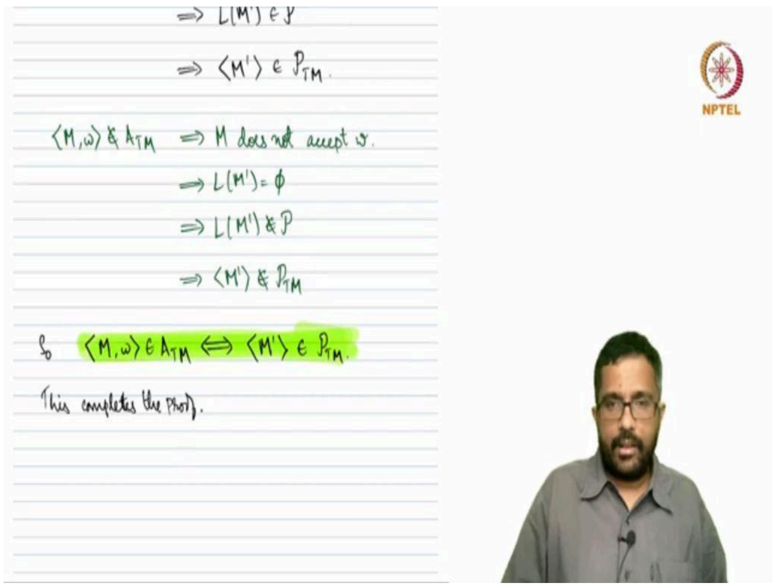
Empty language $\notin P$

That is the case M comma w is a yes instance of A_{TM}. M comma w is an A_{TM}, this is same as M accepting w. In that case when M accepts w, what M' does is to simulate M0 on x. It simulates M0 on x, and does whatever M0 does. So, what is the language now recognized by M'? So, the set of strings recognized by a set of strings accepted by M' is the same as the set of strings accepted by M0, because M this condition is satisfied. So, M' is now simply kind of mirroring whatever M0 does, it does whatever M0 does. So, the language of M' is the same as the language of M0, so that is what I have written here.

Language of M' is the same as the language of M0. But, what is the language of M0? M0 was the Turing machine that was chosen such that it recognizes L0, which is a language that satisfies the property. So, the language recognized by M' is L0. And we know that L0 is a language that satisfies the property, meaning that the language that is recognized by M' satisfies the property, which means M' satisfies the property, which means M' is in P_{TM}. So, the answer to this question does M' satisfy the property P? The answer is yes.

Because the language it recognizes is M, M, sorry L0, and L0 is a language that has a property. So, if M comma w is an ATM, if M accepts w, then M' has the property. So, the YES instance of ATM maps to the YES instance of PTM. Now, we also want to see that NO instance of ATM maps to NO instance of PTM. So, in the case of YES, what happens is if M accepts w, M' becomes a mirror of M0, everything is the same. But, what happens if it is a NO instance?

(Refer Slide Time: 28:46)



$\Rightarrow L(M') \in P$

$\Rightarrow \langle M' \rangle \in P_{TM}$.

$\langle M, w \rangle \notin A_{TM} \Rightarrow$ M does not accept w.

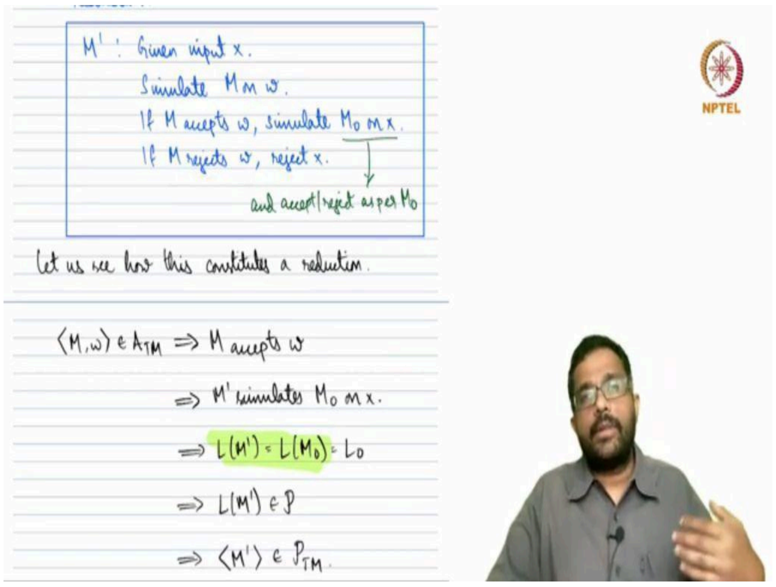
$\Rightarrow L(M') = \phi$

$\Rightarrow L(M') \notin P$

$\Rightarrow \langle M' \rangle \notin P_{TM}$

So $\langle M, w \rangle \in A_{TM} \Longleftrightarrow \langle M' \rangle \in P_{TM}$.

This completes the proof.



```
M' : Given input x.
     Simulate M on w.
     If M accepts w, simulate M_0 on x.  ->  and accept/reject as per M_0
     If M rejects w, reject x.
```

Let us see how this constitutes a reduction.

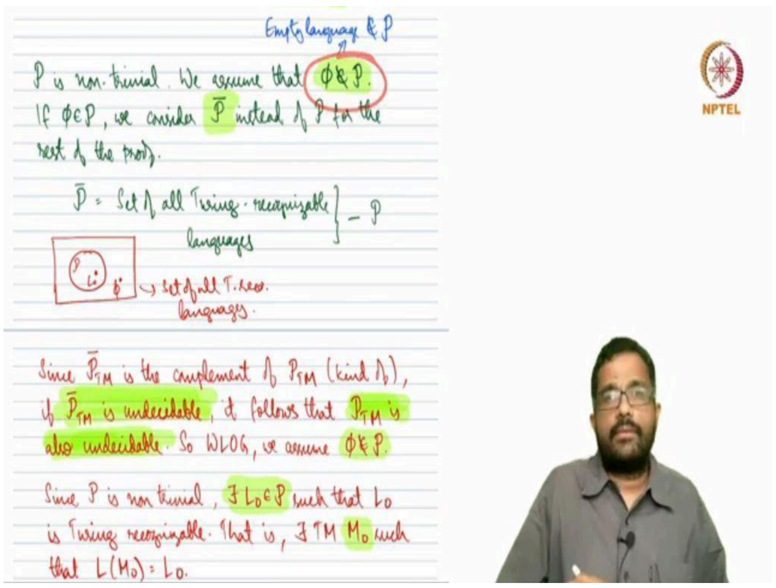
$\langle M, w\rangle \in A_{TM} \Rightarrow M$ accepts w

$\Rightarrow M'$ simulates M_0 on x.

$\Rightarrow L(M') = L(M_0) = L_0$

$\Rightarrow L(M') \in P$

$\Rightarrow \langle M'\rangle \in P_{TM}$.



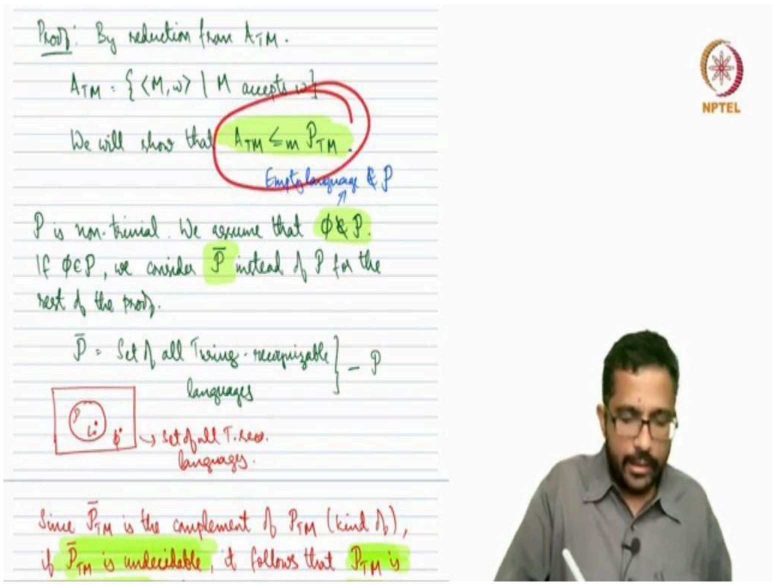
Empty language $\notin P$

P is non-trivial. We assume that $\phi \notin P$.
If $\phi \in P$, we consider $\bar{P}$ instead of P for the rest of the proof.

$\bar{P}$ = {Set of all Turing-recognizable languages} $- P$

Since $\bar{P}_{TM}$ is the complement of P_{TM} (kind of), if $\bar{P}_{TM}$ is undecidable, it follows that P_{TM} is also undecidable. So WLOG, we assume $\phi \notin P$.

Since P is non trivial, $\exists L_0 \in P$ such that L_0 is Turing recognizable. That is, $\exists$ TM M_0 such that $L(M_0) = L_0$.



Proof: By reduction from A_{TM}.

$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

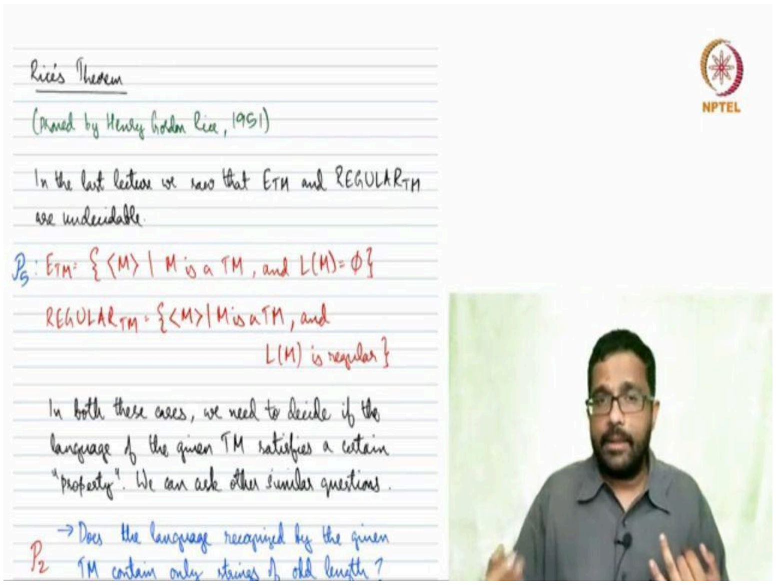
We will show that $A_{TM} \leq_m P_{TM}$.

Empty language $\notin P$

P is non-trivial. We assume that $\phi \notin P$.
If $\phi \in P$, we consider $\overline{P}$ instead of P for the rest of the proof.

$\overline{P} = \left\{\text{Set of all Turing-recognizable languages}\right\} - P$

Since $\overline{P}_{TM}$ is the complement of P_{TM} (kind of),
if $\overline{P}_{TM}$ is undecidable, it follows that P_{TM} is



Rice's Theorem

(proved by Henry Gordon Rice, 1951)

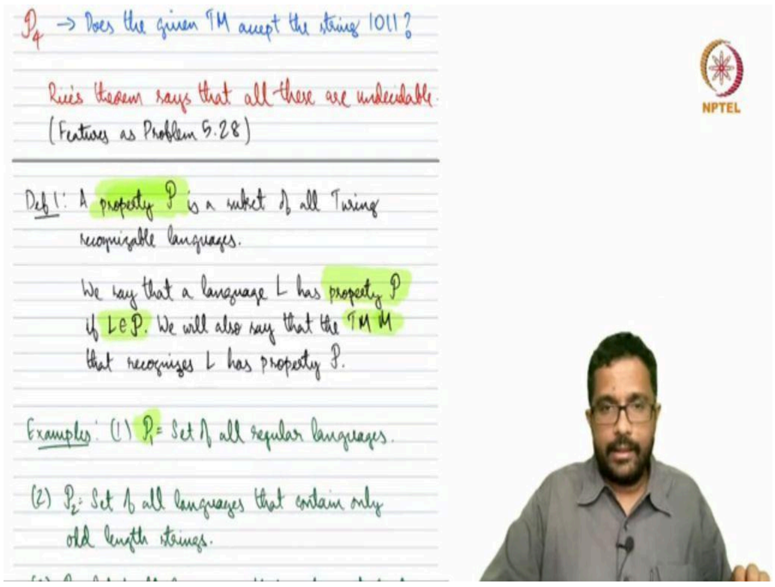
In the last lecture we saw that E_{TM} and $REGULAR_{TM}$ are undecidable.

P_5: $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM, and } L(M) = \phi\}$

$REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM, and } L(M) \text{ is regular}\}$

In both these cases, we need to decide if the language of the given TM satisfies a certain "property". We can ask other similar questions.

P_2 → Does the language recognized by the given TM contain only strings of odd length?



$\mathcal{P}_4$ → Does the given TM acept the string 1011?

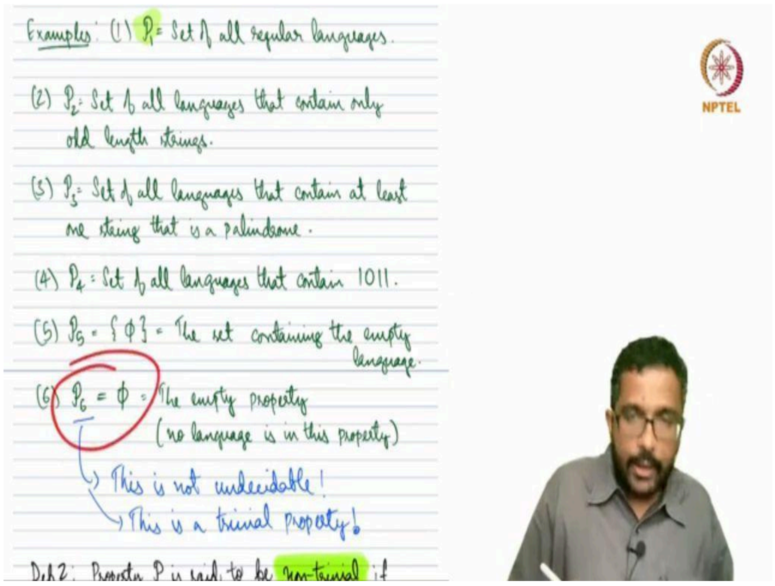
Rice's theorem says that all these are undecidable.
(Features as Problem 5.28)

Def 1: A property $\mathcal{P}$ is a subset of all Turing recognizable languages.

We say that a language L has property $\mathcal{P}$ if $L \in \mathcal{P}$. We will also say that the TM M that recognizes L has property $\mathcal{P}$.

Examples: (1) $\mathcal{P}_1$ = Set of all regular languages.

(2) $\mathcal{P}_2$: Set of all languages that contain only odd length strings.



Examples: (1) $\mathcal{P}_1$ = Set of all regular languages.

(2) $\mathcal{P}_2$: Set of all languages that contain only odd length strings.

(3) $\mathcal{P}_3$: Set of all languages that contain at least one string that is a palindrome.

(4) $\mathcal{P}_4$: Set of all languages that contain 1011.

(5) $\mathcal{P}_5 = \{\phi\}$ = The set containing the empty language.

(6) $\mathcal{P}_6 = \phi$: The empty property
(no language is in this property)

→ This is not undecidable!
→ This is a trivial property!

Def 2: Property P is said to be non-trivial if

Suppose, M does not accept w, so, it could be reject, it could be a loop. So in that case, if M rejects w, M' simply rejects the input, meaning if M rejects w, whatever be the input it does not matter, M' is going to reject it. And if M loops on w, then M' does not even get to do anything, so even then M' does not accept anything.

So, in either case, in either case, the input is not going to be accepted by the machine M'. So, which means the language recognized by M' is the empty language. So, whatever be the input x, it is not going to accept that, because once M rejects w, M' just rejects x, it does not depend on what x was.

So, we are not doing anything if M loops on W also, because if M loops, it just going to run M on w forever. It is never going to get to do anything on x. So, in either case, M' is not going to accept the string x. So, the language recognized by M' is an empty language, and by our assumption an empty language does not have this property. And it was an assumption without loss of generality. So, the language recognized by M' is not in the property, so M' is not in P_{TM}. So, this we have shown that NO instance of A_{TM} if M does not accept W, results in NO instance of P_{TM}.

So, both YES instance and NO instance have been mapped, which means that M, w is an A_{TM}, if and only if M' is in P_{TM}. So, this means that the correspondence of the reduction is has been verified, and indeed this construction is a very fairly simple construction. Suppose, once we are given M and M0, the rest is and w, we can just build this Turing machine. If I are given the descriptions of the Turing machines M and M0 and the string w, it is easy to build this. So, the existence of the Turing Machine M' is not that difficult to see. And we have completed the correspondence and that completes the proof.

So, we have a reduction from A_{TM} to P_{TM}, and that shows that since A_{TM} is undecidable, P_{TM} is also undecidable. So, considering how much we can do with this theorem. So, we are saying that whatever may be the property, it is impossible or it is undecidable, to check whether the Turing machine has a property. And it can you can apply any sort of non-trivial property to it.

So, considering how useful or how broadly applicable this theorem is, the proof is actually not that hard. So, when you consider what we are getting out of this proof it is not that much complex, of course, it is like three four lines and there are simulating two times of simulating going M on w and then M0 and x.

But, still given how many kinds of results we can derive from it or infer from it, it is actually not that difficult, or not considering that it is not that much of an effort. So, that completes the lecture on Rice's theorem. So, what is Rice's theorem? Given a Turing machine, it is undecidable to determine whether the language recognized by the Turing Machine satisfies a certain non-trivial property. So, given a Turing machine it is undecidable to tell whether the language recognized by this Turing machine is the empty language, or the language recognized by this is a finite language, or the language recognized by this contains a certain string.

Or, it contains only certain types of strings, let us say only palindromes or at least one palindrome. All these questions are undecidable. And it is important to have non-triviality. Because if the property is a trivial property like that no language satisfies, then, so for instance, the empty property, no language is going to satisfy this. So, given any Turing machine, you can say it does not satisfy the empty property, because it recognizes some language. And thus, whatever be the language, it is not part of this property, because we know nothing is in this property.

So, it is trivial, it is not undecidable to check whether the Turing machine satisfies a trivial property. There are two trivial properties, one is the empty property and the set of all Turing recognizable languages. So, empty property is not satisfied by any Turing machine, whereas the set of all Turing recognizable languages is satisfied by all the Turing machines. So, one is always no, one is always yes.

Apart from this any other property in between that excludes at least one language, and includes at least one language, all of this is undecidable. So, that is a proof of Rice's theorem, and the proof was by reduction from ATM. So, it involves this and we explained that over detail, and that completes lecture, lecture 41. And in the next lecture, we will see some other techniques for showing undecidability, and that is all for lecture 41. Thank you.