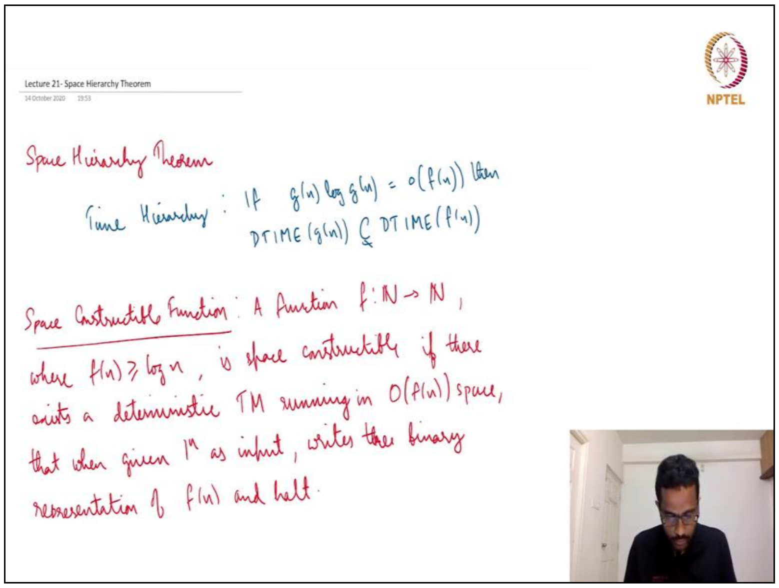
Computational Complexity

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

Lecture -21
Space Hierarchy Theorem

(Refer Slide Time: 00:32)



Lecture 21- Space Hierarchy Theorem
14 October 2020 19:53

Space Hierarchy Theorem

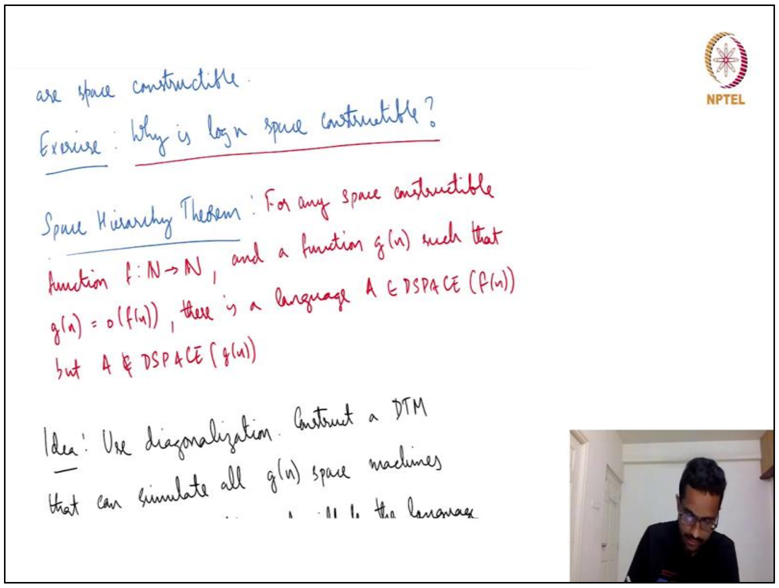
Time Hierarchy : If $g(n) \log g(n) = o(f(n))$ then
$DTIME(g(n)) \subsetneq DTIME(f(n))$

Space Constructible Function : A function $f: \mathbb{N} \to \mathbb{N}$, where $f(n) \geq \log n$, is space constructible if there exists a deterministic TM running in $O(f(n))$ space, that when given 1^n as input, writes the binary representation of $f(n)$ and halt.

Hello and welcome to lecture 21 of the course computational complexity in this lecture we will see space hierarchy theorem. So, in lecture 13 we had seen time hierarchy theorem and let me just recap time hierarchy theorem basically said that if you have more time then you can compute certain languages or problems that you could not do if you did not have that more time. So, having more time certainly helps that was the that is what we could infer from the time hierarchy theorem.

To be more precise let me just write once again to be more precise it said that if gn log gn was little o of fn then D time gn is contained strictly contained in d time fn. Basically if you have more time then you could compute certain functions or certain languages which you could not compute if you had the less time. So, space hierarchy theorem is going to be very similar just the time is going to be replaced by space. And even the proof technique that we are going to use is going to be very similar.

(Refer Slide Time: 01:58)


are space constructible.

Exercise: Why is log n space constructible?

Space Hierarchy Theorem: For any space constructible function $f: \mathbb{N} \to \mathbb{N}$, and a function $g(n)$ such that $g(n) = o(f(n))$, there is a language $A \in DSPACE(f(n))$ but $A \notin DSPACE(g(n))$

Idea: Use diagonalization. Construct a DTM that can simulate all $g(n)$ space machines

[illegible] the language

So, if you may recall that we had defined time constructable function. Similarly for the time constructable function we defined for the time hierarchy theorem. Similarly we will define a space constructable function. So what is the space constructable function? Basically it is a function fn such that fn itself can be computed in fn space. So, the space required to compute the function is equal to the is it is at most the function itself.

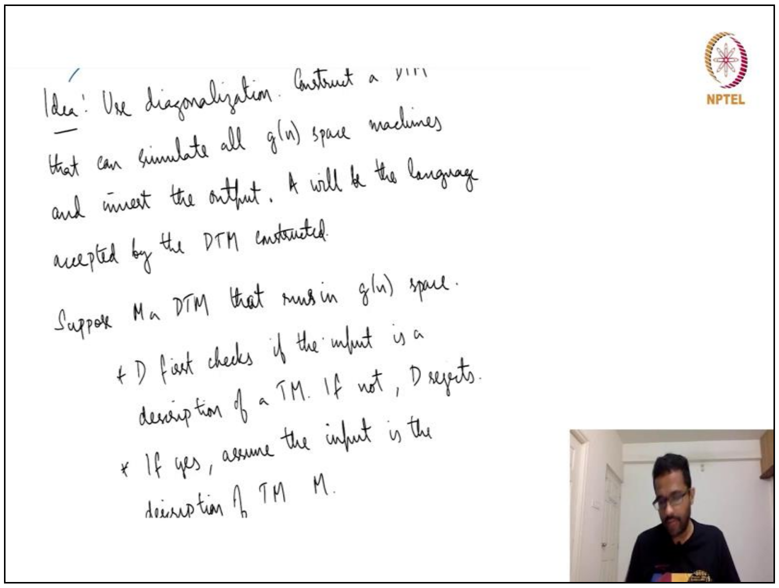
And a bit more technical detail is that the input should be given as a unary string as a string of ones the number of ones being the being the input being the input n and the output should be the actual function. Because we have to write in unary the input because because usually the input length is always n. So, here we want to write the input in unary. So, this is this definition of space constructability.

Basically a function fn should be construct should be computable in the time in the space bound given by the function itself. Time constructable was exactly the same just that you had to replace this space bound with a time bound and like I said for time constructable functions most of the familiar function that we have like n squared n log n square root n log n everything all of this is space constructable.

You will have to think of really obscure functions to get something that is not space constructable. So, maybe one exercise that you can try out y is log n space constructable can you work compute can you see how to compute given n or given n once in a string how can we compute log n here in using at most log n space. You can try you can try that. So, now let me formally stray state the space hierarchy theorem. So, for any space constructable function f there is sorry and a function g such that gn is little o of fn.

There is a language A that is in DSPACE fn sorry button but not in DSPACE gn. So, fn is a time constructable sorry space constructable function and consider a function that is little of fn meaning little o means limit gn divided by fn tends to 0 as n goes to infinity. So, gn grows this at a strictly smaller base than fn limit gn divided by fn goes to 0. In such a case there is always a language that can be decided in fn space but cannot be decided in lesser space than that in gn space.

(Refer Slide Time: 06:11)



Idea: Use diagonalization. Construct a DTM that can simulate all $g(n)$ space machines and invert the output. A will be the language accepted by the DTM constructed.

Suppose M a DTM that runs in $g(n)$ space.

- D first checks if the input is a description of a TM. If not, D rejects.
- If yes, assume the input is the description of TM M.

So, just to formally or to state it in another way. So, if gn is lit low of fn and fn is time space constructable then DSPACE of gn is contained but strictly contained in DSPACE of fn. So, this containment is strict this is what space hierarchy theorem means. So, just to contrast this with time hierarchy it is exactly the same just that in the time hierarchy theorem we required gn log gn to be little o of fn over here we just need gn to be little o of fn.

So, there was a gap of log n and that was that was required in the case of time hierarchy theorem in the case of space hierarchy theorem we do not need that. So, even things that are very close very, close closer than log n but as long as they are lit low of fn it is fine. So, for instance if like let us say fn is n squared or let us say gn is n squared and fn is let us say n squared square root of log n.

Even in the in this case DSPACE of gn is strictly contained in the space of fn whereas in we cannot use the same g and f for time hierarchy theorem because this does not have a gap of log n this g and this f we need a gap of log n which is not there for this g and this f we cannot use these two functions to get a gap in the case of time hierarchy theorem. So that is a statement what is a proof idea? The proof idea is also this is also idea is also very similar to that of time hierarchy theorem.

So, I will present it slightly differently. So, that. So, that maybe sometimes when you see multiple or slightly different way of presenting the same thing it helps the understanding better. So, you may recall that in the case of time hierarchy theorem we used diagonalization. So, we first saw the proof of the halting problem or the acceptance problem and then we saw how to modify that how to adapt that for the time hierarchy theorem.

So, here since we only since we had only recently recapped that for the time hierarchical theorem I am not going to recap the proof of the haunting problem but will just use the same but then the techniques are exactly the same. So, what did we do in the case of time hierarchy theorem we basically had a machine that simulates another machine. So, if you recall we wanted to show the separation between n and n squared.

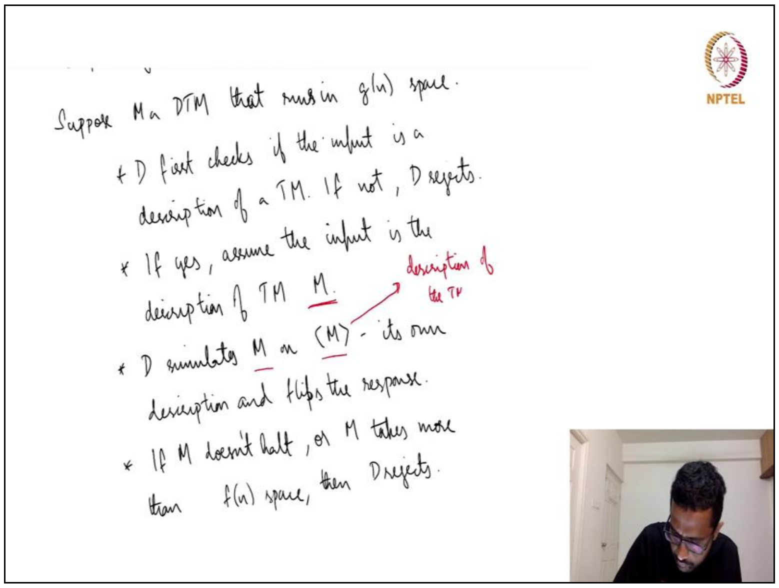
So, basically we would simulate an n machine with in n square time sorry n power 1.9 time. So, this is what we had. So, whatever input we get the input being a turing machine we just simulate the machine on its own description for endpoint and power 1.9 steps and then reject and flip the output. So, basically we simulate the machine on its own description and then flip the output while making sure the time was at most n power 1.9.

In the case of; so, here we will one change in the presentation is that here we will stick to abstract functions and not specific functions like that because we have already seen it. So, maybe

I would like everybody to comfort to be comfortable with this abstract function. So, we will not use concrete functions and second point is that we will there we wanted to show that the we wanted to run for n power 1.9 steps which is something less than n squared but more than n.

So, here we will try to do the same thing but with space. So, we will make sure that it it runs it takes adequate space but no more than fn. So, we will try to simulate a gn space machine with an fn space machine but we will we will have to curtail the space usage. So, that brings in a small difference on how this implementation is done.

(Refer Slide Time: 11:24)



Suppose M a DTM that runs in $g(n)$ space.

* D first checks if the input is a description of a TM. If not, D rejects.
* If yes, assume the input is the description of TM $\underline{M}$.
* D simulates $\underline{M}$ on $\underline{\langle M \rangle}$ - its own description and flips the response.
* If M doesn't halt, or M takes more than $f(n)$ space, then D rejects.

And again to answer the earlier question that I myself had asked in the case of time hierarchy theorem we have this login factor this one which is not appearing. In the space hierarchy theorem there is no log in here why is that if you if you recall in the time hierarchy theorem we had used a certain result that to simulate any turing machine with its description by Henny and Sterns which said that any turing machine that runs at time t needs to be needs t log t time to simulate by another turing machine by a universal turing machine.

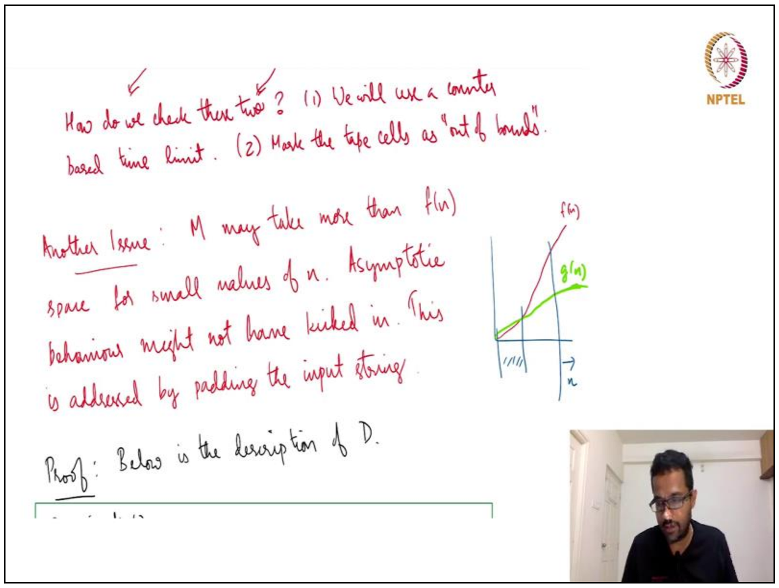
But however in the case of space restricted turing machines we could we can we can simulate it but and there is no need of this extra log factor. So, that extra log factor does not appear in the equivalent simulation of the space bounded machines and that is this extra log factor that we are saving in the space hierarchy theorem. So, now let us get on to the idea the main idea is to construct a deterministic turing machine that simulates all the gn space machines and flips the output and then flips the output.

So, and whatever is a language accepted or recognized by the turing machine that we construct that will be the language that that is in DSPACE fn but not there in DSPACE gn. So, basically what we do is we first check whether the machine first checks whether the in; so, the input is read as an description of a it is interpreted as a description of a turing machine. So, you could describe that turing machine.

So, is it a description is first checked this is something that is straightforward are there states are there transitions and. So, on if it is a description if it is not a description it is immediately rejected if it is a description of a turing machine M say if it is a description of M then what it does is it simulates the machine M on its own description. So, if it is a description of a turing machine. So, then D simulates the machine m. So, on the description of m, so, this is the description of the turing machine.

So, this is used to this M with angular brackets is denoted to you is used to denote the description of the turing machine m. So, it simulates M on an input. So, the input could be anything. So, here we are simulating it on its own description and then flip the output and there are some things that can happen.

(Refer Slide Time: 14:09)


How do we check these two? (1) We will use a counter based time limit. (2) Mark the tape cells as "out of bounds".

Another Issue: M may take more than f(n) space for small values of n. Asymptotic behaviour might not have kicked in. This is addressed by padding the input string.

Proof: Below is the description of D.

One is that the machine may not halt but how do we deal with that because it and second is that it may take more than the allotted SPACE. So, we want. So, we want d to be an fn space bounded machine. So, these two things have to be prevented it is not halting and it taking more space. In the case of time hierarchy theorem we stopped the simulation at a certain point right because we wanted the time to be bounded but here it is not a time bounded machine.

So, how long do you keep simulating. So, this does not hard problem how do we stop it. So, we will address both. So, what we will do is. So, the question is how do we check these two this end and this uh. So, one is. So, one meaning this one the does not halting not halting um. So, we will sorry we will use a counter based sorry counter based time limit and for two basically we want to constrain the space usage. So, we will just mark some cells that's unbounded like as as you cannot as out of bounds. So, mark the spell cells tip cells as out of bounds.

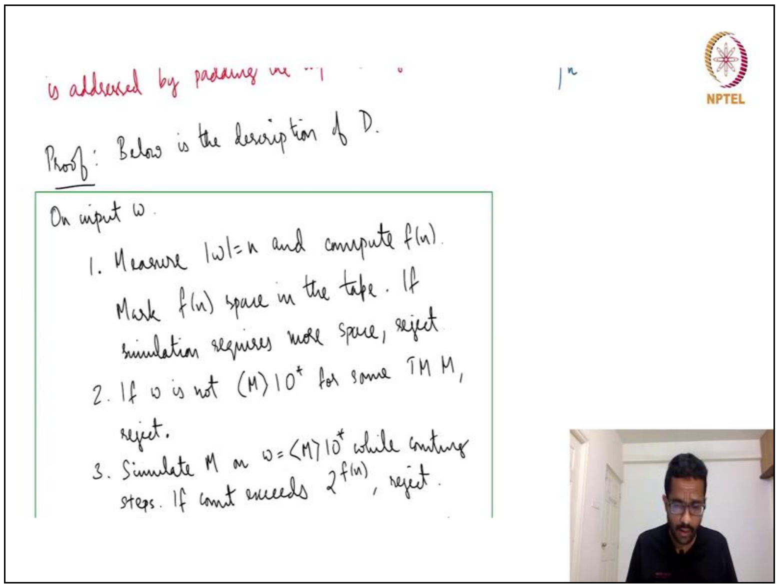
So, basically we say which is the territory that the machine can operate and there is another. So, these two handles these two these two solutions handle these two issues the another thing is that this had this was also there in the case of time hierarchy but I perhaps I did not elaborate to this detail. So, as to not cause any confusion because I wanted to make it essentially the diagonalization.

So, now that we have seen diagonalization I just want to bring out this particular issue. So, one is that even though fn is gn is little of fn it could. So, happen that in for small values gn could be higher. So, what I am saying is. So, the graph could be something like this. So, this could be maybe the red is fn and gn could put not like this but maybe something like this let us say this is gn green is gn but for small values.

So in this case it could it is possible that fn gn is lit low of fn maybe what I have drawn look like a constant but maybe there is this needs to taper off slightly something but the point here is that in this region in this region gn is greater than fn in this shaded region. So but that is ok because all that the asymptotic says that if g n is little of f n it is beyond a point this should happen gn should be less than some constant times fn for any constant. So, we will choose.

So, we will choose or we will make sure we will make sure that the n that we have we have to make sure that the end that we have is beyond this place it is sufficiently large maybe the end that we have is restricted to this point or something beyond this big beyond this most line because there has to be sufficient gap between f and g. So, we will choose the n such that it is sufficiently far away to the where the asymptotic effect has happened. So, these are the three issues and these are how they are addressed.

(Refer Slide Time: 18:53)



is addressed by padding the [illegible] 1^n

Proof: Below is the description of D.

On input w.

1. Measure $|w| = n$ and compute $f(n)$. Mark $f(n)$ space in the tape. If simulation requires more space, reject
2. If w is not $\langle M \rangle 10^*$ for some TM M, reject.
3. Simulate M on $w = \langle M \rangle 10^*$ while counting steps. If count exceeds $2^{f(n)}$, reject.

So, now let us see the description of the turing machine or more formal description of the turing machine of the machine D whose the language the language that is in DSPACE fn is the language recognized by D DSPACE f n but not in DSPACE gn. So what is D? So, given an input W you measure w measure the length of W. So, M length of w is let us say it is n and compute fn . So, you compute fn and this is why we need fn to be space constructable because we want to mark out fn space.

So, for that we need to know what is fn? So for that we need to compute fn but then we only have fn space that's why we need to we need fn to be space constructable. So, we mark fn space in the tape and if the simulation at any point crosses this fn. So, you you put a marker or something if it crosses that marker you immediately reject. So, anything that you compute this computation will not cross that marker that is step one.

And now you look at the input. So, if the input is not of the form the description of a turing machine followed by 11 and a string of 0's. So, this is 0 star 0 star means it could be 0 0 0 0 or something if it is not of the form this form a description of a turing machine one one and a bunch of 0's you reject. So, it has to be a description of a turing machine 1 1 and a bunch of 0's. So, the bunch of 0's could be 0 0 or 1 0 or many 0's more than one 0s.

Now you simulate the machine M ok. So, there is a description or turing machine M stands for the description of a turing machine. So, the same M you simulate the machine M on w. So, w is M followed description of M followed by one and the mini 0s and see and while keeping track of how many steps you make if the number of steps exceeds 2 power fn we reject why do we do that this is because we want to be more than any gn space bounded machine.

So, any gn space powered machine will have 2 power constant gn number of configurations and therefore that mean number of steps. So, while 2 power fn, fn is bigger than gn sufficiently bigger than gn. So, 2 power fn will be sufficiently bigger than 2 power c constant times gn that is why we stop the number of steps at 2 power fn. So, otherwise how will we know if it continues running.

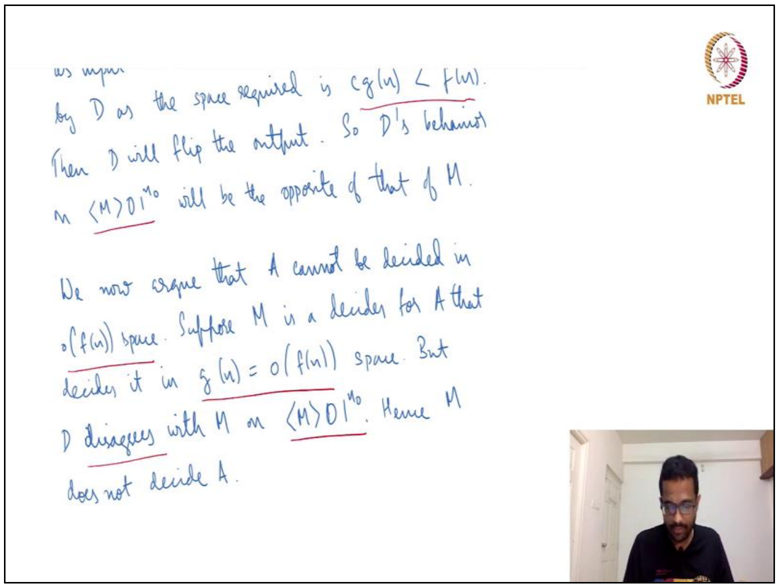
So, if the count exceeds this you reject now what you do finally is to flip the output. So, this is just flipping of the output if M accepts then you reject and if M rejects then you accept. So, now this may seem a bit cryptic but all that we are having. So, which is slightly different from the time hierarchy theorem proof is that we have this description of M followed by one and a bunch of 0’s. So, and then M is simulated in time hierarchy theorem we simulated M on its own description but here we simulated we are simulating M on the input string which is the description followed by some bunch of one and 0’s.

So, this 1 1 and 0s the role it plays is to make the length sufficiently large. So, that we are in this most region that is the role that it plays and then we flip the output and step three step three is making sure that it doesn't run too many steps. So, anything that runs in gn space can be decided in gn space will not take more than two power fn steps now what is what is the language that is in DSPACE fn but not only space gn it is a language of this machine a now clearly what is the space usage of d in order to simulate.

So, first one point is that M is being simulated by d so we may request. So, the alphabet of M has to be written in terms of alphabet of D it may or may not be the same. So, that brings in a constant overhead constant overhead. So, c times the space usage of M may happen. So, this is and this is the main overhead if M uses gn space D will use constant times gn space where the constant is denoted by the letter c and in the case of time hierarchy theorem we had a log log overhead for the simulation for whereas in the space hierarchy theorem the overhead is just a constant.

And then this is what I said earlier we want to make sure that we only consider large enough n. So, what we do for that we will choose an n where. So, this constant c times gn this is less than fn. So, we will choose a constant or we'll choose n such that c times gn is less than f n for all the n above this value. So, we will choose all the n to be above n 0 where sorry where n 0 is this cut off point where g n is or c times g n is g n is sufficiently smaller than fn or c times gn is also still less than fn. And it continues to maintain this can this continue to maintain beyond that point.

(Refer Slide Time: 25:10)



as input
by D as the space required is $cg(n) < f(n)$.
Then D will flip the output. So D's behavior
on $\langle M \rangle 0 1^{n_0}$ will be the opposite of that of M.

We now argue that A cannot be decided in
$o(f(n))$ space. Suppose M is a decider for A that
decides it in $g(n) = o(f(n))$ space. But
D disagrees with M on $\langle M \rangle 0 1^{n_0}$. Hence M
does not decide A.

There will be such an fn because gn is little of fn. So, this is to make sure that the asymptotic effects effect has kicked in. So, that takes care of the details. And now let us see what happens the final contradiction. So, what happens now when the machine D is fed a machine M the description of a machine M followed by 0 1 and 1 repeated n 0 times. Now in this case what will; So, now D is not M is any machine.

So, M is a description of a machine. So, the first step will first check first step will pass this will pass and the length is n 0 plus 1 plus the description of M and it will mark sufficient space. The space required for this will be fn and fn we know we know that it is more than n 0. So, n is at least n 0. So, we have at least n 0 ones. So, f of n will be at least c of c times g of n. So, M has enough space to successfully simulate sorry D has enough space to successfully simulate m.

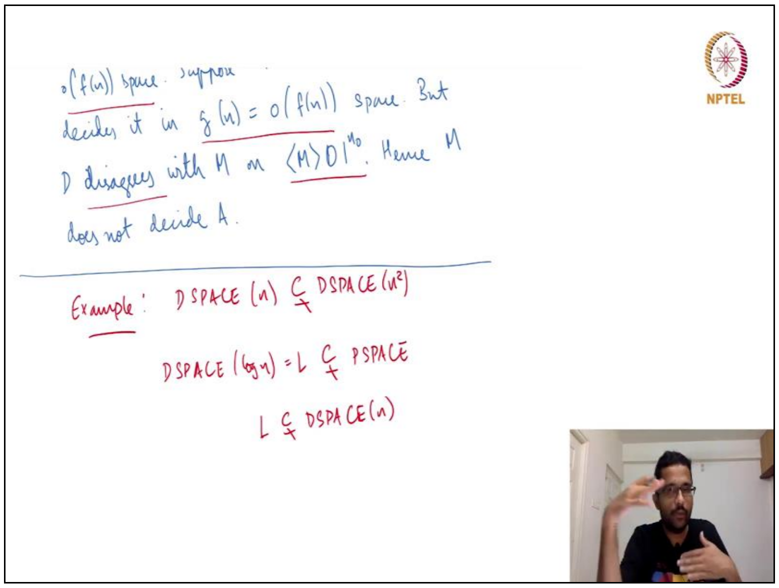
So, D can simulate M and then flip the output. So, D when it is fed m; description of M 0 1 power n 0 will be opposite of whatever M does on its own description or M does on the same input M description of M followed by 0 followed by 1 power n 0. This is because d has the time and space to successfully simulate M because the time the space bound and the time limit is chosen such that the input like the students at the space bound and time limit are respected.

Now A cannot be decided in little of fn space this is the contradiction why is this? Suppose M is a decider for suppose of course D takes order fn time sorry space. Now suppose there was another decider M that uses little o of fn space let us say it uses g in space let us say M is a decider for a that uses gn space but we already said that. Now what happens when d is fed description of M followed by 0 followed by 1 power n 0 description of M followed by 0 followed by one power n 0 is fed to D.

So, what we just saw is that D will do whatever the opposite of whatever M does on this . So, but then now M is supposed to be a decider for the same language as D but then we just saw that B will do whatever the opposite of whatever M plus this is a contradiction. So, M and D disagree on this input. So, that is a contradiction. So, M cannot be a decider for the same language as D which means D has to be D yeah which means there is a the language decided by D which let us call it a cannot be decided in little o of fn space.

So, we get the statement of the time hierarchy space hierarchy theorem that DSPACE gn is strictly contained in DSPACE fn strictly means it is subset but not a not equal to DSPACE of f n whenever gn is a little o of fn.

(Refer Slide Time: 29:09)



$o(f(n))$ space. Suppose

decides it in $g(n) = o(f(n))$ space. But

D disagrees with M on $\langle M \rangle 0 1^{n_0}$. Hence M

does not decide A.

Example: $DSPACE(n) \subsetneq DSPACE(n^2)$

$DSPACE(\log n) = L \subsetneq PSPACE$

$L \subsetneq DSPACE(n)$

Once again why is it called hierarchy theorem because it define defines a hierarchy in the space usage for instance. So, let us say fn is gn is n and fn is n squared then we get the DSPACE of n is strictly contained in DSPACE of n squared. And let us say maybe DSPACE of login what is this is nothing but l is strictly contained in the space of Ii do not know or maybe y naught PSPACE in fact in fact L is actually contained in it is even contained in the space of n itself. So, the space of n is contained in PSPACE.

So, you could say this as well. So, basically there are for each for any two functions f and g if g is asymptotically smaller than f then there is something that is decided by in f space but not in g space. Just a short quick recap of the main idea of the diagonalization the language that is decided in f space but not decidable in g space is not explicitly defined but by through the construction of a turing machine.

So, by construction this turing machine uses at most fn space. So, basically it uses it it marks off fn space and then it has a counter of that goes up to two power fn but both of them use fn space. And then what it does is what we make sure is that the input is of sufficiently long length. So, that the asymptotics have kicked in. So, that any input of length gn can be simulated in this space and. So, we first see that any turing machine of length that that uses gn space when you feed its own input to this machine.

So, when you feed the input M 0 or sorry M 1 followed by n 0 0's to D, D does the opposite of whatever M itself is supposed to do on its on the same string suppose this W. So, D what D does one W is the opposite of what M does on W because D is programmed to do that because D simulates M on this and then does the opposite and. So, now what if M decides the same language as D using lesser space than D then by the same argument D disagrees with M on this input which means M cannot be deciding the same language as D and that is a contradiction.

So, we assume that there is a there is an machine M that decides D using lesser space that assumption was incorrect and that summarizes the proof of space hierarchy theorem. So, just like we had time hierarchy theorem we also have space hierarchy there. So, you can do more things with if you have more space that you could not have done with done if you did not have the extra space, thank you.