

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

Lecture -24
Polynomial Hierarchy Using Oracles

(Refer Slide Time: 00:15)

NPTEL

Exist: If $A \in P$, then $P^A \subseteq P$

PH using oracles

We can define classes in polynomial hierarchy (PH) using oracles.

We will see that $\Sigma_2^P = NP^{SAT}$

In general, $\Sigma_{i+1}^P = NP^{QSAT_i}$ and $\Pi_{i+1}^P = Co-NP^{QSAT_i}$

Theorem: $\Sigma_2^P = NP^{SAT}$

Hello and welcome to lecture 24 of the course computational complexity. In the previous lecture we saw oracle turing machines in this lecture we will see how we can view the polynomial hierarchy that by that I mean the classes in the polynomial hierarchy such as sigma 2 sigma 3 pi 2 pi 3 etcetera you can you can get them. So, we define them as with this alternating quantifiers. So, now there is another way to arrive at these classes in the polynomial hierarchy using the definition of oracle turing machines. So, we will see how to get that.

So, we can define these classes using the oracle turing machines what we will see in this today's lecture is that sigma 2 is the same as the class NP with access to a SAT oracle. Sigma 2 is equal to NP with access to a SAT oracle. So, let us see how we will see that we will see the detailed proof and in general what is true is that sigma i is NP with access to a q SAT i oracle. So, you may recall that q SAT i was the complete problem in the complete problem of the sigma sigma i.

So, Σ_1^P is the Σ_1^P SAT is a complete problem of Σ_1^P . So, Σ_2^P is the there are these alternating quantifiers. Similarly I am just trying to see if I defined it I think I just defined it to be similar to Σ_2^P but Σ_1^P is a complete problem for the class Σ_1^P . So, Σ_{i+1}^P is NP with access to a Σ_i^P oracle. So, basically if you have it is NP with access to the complete problem of the of the of the high of the class just one level below.

So, here at Σ_{i+1}^P this the here at Σ_{i+1}^P for that we need NP with access to Σ_i^P complete language and similarly Π_i^P is co-NP with access to a Σ_i^P complete problem Σ_i^P which is Σ_i^P complete problem. So, this is ah. So, these results are the proof of these lessons are exact exactly like the same way that we will see the proof of Σ_2^P is NP set. So, because of which we will not really be doing the other proofs we will just show the proof that Σ_2^P is NP with access to a SAT oracle.

(Refer Slide Time: 03:18)

Theorem: $\Sigma_2^P = NP^{\Sigma_1^P}$.

Proof: $(\Sigma_2^P \subseteq NP^{\Sigma_1^P})$

$L \in \Sigma_2^P$ iff $\exists v \forall w \exists x$ poly time $V, s.t.$

$x \in L \Leftrightarrow \exists y \forall z, V(x, y, z) = 1.$

Consider such an L . We will show how to construct an NTM N that runs in poly time with access to SAT oracle and decides L .

So, like I said in the lecture 23 the previous lecture whenever we want to show something like a language or class A is equal to class B we end up showing containment in both directions. So, we will show that left hand side is contained in the hand side and vice versa. So, like that we will let us do first direction that Σ_2^P is contained in NP with access to a SAT oracle, so, which is what we have here. So, and again how do we show that some, one set is contained inside the other again the standard trick is to pick an arbitrary element of the the first set.

So, in this case sigma 2. So, let L be a language in sigma 2 and now we will show that this L is in NP with access to a SAT oracle. So, an arbitrary member here if we can show it to be there then it means that any member here is there and. So, this is a subset of that. So, we will take L to be in sigma 2. So what does it mean when I say that L is in sigma 2 when I say L is in sigma 2 this means that L has a there is a polynomial deterministic polynomial time verifier P such that whenever x is in L whenever x is in L there are strings y or there exists a string y such that for all z the verifier evaluates x y z to 1.

Again there are other things about y and z being a polynomial length I am just skip that detail. So, this is the definition of sigma 2, L is in sigma 2 if there is a polynomial time verifier y V such that for all x and L it must be the case that there is a y such that for all z, V x y z evaluates to 1 or the deterministic verifier successfully verifies x, y and z.

(Refer Slide Time: 05:23)

Construct an NPM N with access to SAT oracle and decide L .

NPM N : Given input x , it will guess y , and then needs to decide if $\forall z V(x, y, z) = 1$.

Once x, y are chosen, we can view $V(x, y, z)$ as $V_{x, y}(z)$ → function of z alone.

$\forall z V(x, y, z) = 1 \iff \forall z V_{x, y}(z) = 1$

The slide also features the NPTEL logo in the top right corner and a small video inset of the presenter in the bottom right corner.

So, now consider such an L now how do we now we need to show that such an L is also there in NP with access to a SAT oracle. So, what can that? So, given so, the n, so, basically we have to construct the NP machine the non-deterministic turing machine non-deterministic polynomial time turing machine with access to a SAT oracle. So, let us see how to do that? So, the non-deterministic turing machine, so, basically given input x it can guess the y it can guess this y which is a which is because NP has a capability to guess.

But NP cannot do this for all z thing. So, NP cannot do that because NP can only guess something if it is there. So, basically given input x it can guess y and now given x and y are fixed then it needs to decide if this is true what is true for all z is $\forall x y z$ equal to 1 or does the verifier accept the triplet $x y z$. And slight notation change may help to change the perspective and view things differently. So, once x and y are fixed, so, x is the input and y is the guess ones y is guess. Now $\forall x y z$ is just a function of z .

So, I may one thing that we can do is to bring down x and y as a subscript and then view it as a function of z alone. So, once x and y are fixed we can view $\forall x y z$ as a function of z alone which is what is happening here function of z alone.

(Refer Slide Time: 07:06)

The slide contains the following handwritten content:

- Top right: NPTEL logo
- Equation 1: $\forall z V_{x,y,z} = 1 \iff \forall z V_{x,y}(z) = 1$
- Equation 2: $\iff \text{Neg}(\exists z V_{x,y}(z) = 0)$
- Equation 3: $\iff \text{Neg}(\exists z V'_{x,y}(z) = 1)$ (circled in red)
- Note: where $V'_{x,y} = \text{Flip the output of } V_{x,y}$
- Note: can be decided by a SAT oracle

- Equation 4: $\text{NP}^{\text{SAT}} \subseteq \Sigma_2^P$
- Text: Consider a language $L \in \text{NP}^{\text{SAT}}$. L is decided

A small video inset in the bottom right corner shows a man in a yellow shirt speaking.

So, now the NP machine needs to decide again notice that till now we have not made use of the SAT oracle NP machine needs to decide whether for all z , $\forall x y z$ is equal to 1 which is the same as we defined $\forall x y z$ to be this $\forall x y$ with $\forall x y$ as subscript of z and. So, what is the negation of this? The negation is there exists z such that $\forall x y z$ equal to 0 for all z \forall is equal to 1 and the negation is that there is some z for which it is 0 for all z something happens for one z it does not happen that is a negation.

So, negation is there x is z for which $\forall x y z$ is 0. Now instead of $\forall x$ I can change the verifier I can I can modify the verifier to give another polynomial time verifier which just flips the output.

So, I define a new verifier called V' and V' is just flip the output of V . So, you just what let V run and you flip the output. So, we suggest a flip. So, $V = 0$ means $V' = 1$ that is what we have in this in this red circled input.

So, we want the negation of this. So, is the opposite of this Σ_2 is what we want know but does there exist a z such that V' is V of z is equal to 1 is just a satisfiability query. So, you have a verifier which can be viewed as a formula also. So, you can the entire verifier is an algorithm which like in Cook Levin theorem you can do it as a formula and you are asking whether there is a z that satisfies it you are asking whether this is z that satisfies.

It if there is a z that satisfies it. So, that is a satisfiability query. So, you can ask the satisfiability query and what we want is the negation of that. So, the negation is just you flip the output again. So, all that it boils down to is a negation query. So, that is what we need to do here it is a negation query it is a negation of a satisfiability query. So, just once again we need to have given a language in Σ_2 we need to view it as we need to see how it is in NP with access to SAT oracle.

So, Σ_2 means it can be written in this form with the polynomial time verifier V there x is y says that for all z , $V(x, y, z) = 1$. So, then NP and N can guess y and it needs to decide for all z , $V(x, y, z) = 1$. So, instead of doing that it we can we realize that we can view it as a satisfiability query does that exist as z that such that something happens and that can be easily be done because we have access to a SAT oracle.



So, we just make the access we just make the query to the SAT oracle and then decide accordingly. So, that is why Σ_2 is contained in NP with access to SAT oracle. So that is one direction of the proof the other direction is pending.

(Refer Slide Time: 10:51)

NP.

N may query the SAT oracle many times during its execution.

Suppose $x \in L$, then there is a sequence of non-deterministic choices c_1, c_2, \dots, c_m made by N, and answers to oracle queries $a_1, a_2, \dots, a_k \in \{0, 1\}$, such that N makes the above choices, receives the above answers (a_i to i^{th} query) and accepts x .

What is the other direction that NP with access to SAT oracle is contained in Σ_2 . So, again the same thing, so, we take an arbitrary language from the left hand side which is NP with access to a SAT oracle. So, which means L is decided by a non-deterministic polynomial time machine n with an oracle access to satisfiability. So, this is this direction is slightly more involved than the other direction. So, let me just go a bit slow.

So, what can n possibly do in is in a deterministic turing machine. So, which means it has possibly multiple computation paths multiple acceptors x outcomes and it is also it also has access to a sad oracle which means it will keep it will query the SAT oracle from time to time it can query the SAT oracle from time to time and it can also use this in an adaptive manner. So, you can make a query and then if the oracle if the response is yes then it does something if it does sponsors no all this is part of the algorithm.

So, these are the things that N is capable of doing. So, it is a non-deterministic machine not NP with access to a SAT oracle suppose. So, just to summarize or just to make say it very formally suppose x is in L that means it is an orthodontic machine. So, that means there is one sequence of choices that leads to x being accepted because it is not determinism. So, let these choices be c_1 to c_m .

It will be a fixed number of choices for a fixed input. So, it let it make m choices by the non-deterministic turing machine N . And in the in this process, so, this process will guide it using some computation paths. So, it is it has access to the SAT oracle. So, it may be making many questions queries to the SAT oracle. Let us say it makes k queries where k is another fixed number k queries to the SAT oracle let the responses to the k queries be a_1 to a_k .

These are all yes no responses equivalently we will view them as 0 one responses this is yes in sense this is no instance. So, for each of the queries it receives a_i as a response. So, what it does is? ah. So, I am just trying to represent what it does.

(Refer Slide Time: 13:54)

let ϕ_i denote the i^{th} query to the SAT oracle.
 If $a_i = 1$, then $\exists u_i$ an assignment such
 that $\phi_i(u_i) = 1$
 If $a_i = 0$, then \forall (assignments) v_i , we have
 $\phi_i(v_i) = 0$.

So $x \in L \iff \exists c_1, c_2 \dots c_m, a_1, \dots, a_k,$
 $u_1, u_2 \dots u_k \forall v_1, v_2 \dots v_k$
 such that N accepts x using
 choices c_1, \dots, c_m , and answers

In a pictorial fashion, so it makes many choices c_1 to c_m and queries it makes queries ϕ_1 , ϕ_2 etcetera ϕ_k . So, I said it makes k queries. So, and recall it is each of them is a instance of satisfiability because that is the oracle that we have. So, let the query. So, every time it must be writing a SAT instance what is that instance it is a Boolean formula that the query. So, the Boolean formula instance let us let it be ϕ_1 , ϕ_2 ϕ_3 up to ϕ_k and the responses answers are a_1 a_2 up to a_k , k queries k answers.

And if x is in L there is a sequence of choices c_1 to c_m such that which makes some k queries and which makes some corresponding to the queries you get the correct answer. So, again the one the queries are fixed by the choices and the answers are decided by the queries it s in sensor

no instance is the oracle does not lie. So, it just gives the correct answer. So, this is what happens and ah. So, again this is just what I have written here if i denote the i th query.

So, if the answer is 1 which means it is satisfiable ϕ is satisfiable which means that there is a satisfying assignment for ϕ let it be let us call it u . So, u is a satisfying assignment for ϕ which means ϕ or sorry ϕ I said u its u . So, ϕ when given the assignment u it is it evaluates to true or one if a_i is 1 if a_i is 0 which means it is not satisfiable which means there is no satisfying assignment which means whatever assignment you try whatever assignment V_i you try for all assignments V_i ϕ will be 0.

So, whatever you input it will be 0 again what was the goal here? The goal here was to show that L is in Σ_2 we started by saying L is an NP with access to SAT oracle which means it makes non-deterministic choices and gets yes no answers and we are trying to interpret what these yes no answers are. So, if yes answer there is a satisfying assignment to the query if it is a no answer there is no satisfying assignment to the query that is where we are.

Now all that we need to do now. So, till now it seems like we were just understanding what it means for this language L to accept a string x it needs to have a sequence of choices and then queries and then yes and no and all that. Now all that we need to do is to write these things in a formal logical sense and that will start looking like a Σ_2 language already. So, let us see.

(Refer Slide Time: 17:38)

NPTEL

$\phi_i(u_i) = 0$

So $x \in L \iff \exists [c_1, c_2, \dots, c_m, a_1, \dots, a_k]$
 $u_1, u_2, \dots, u_k \quad \forall [v_1, v_2, \dots, v_k]$

$\phi_i(u_i) = 1$ when $a_i = 1$ } such that N accepts x using choices c_1, \dots, c_m , and answers a_1, \dots, a_k , and

$\bigwedge_{i=1}^k \left[(a_i = 1 \Rightarrow \phi_i(u_i) = 1) \wedge (a_i = 0 \Rightarrow \phi_i(u_i) = 0) \right]$

So, all I am doing is to just write it very formally in a logical step sense. So, if x is in L or x is in L if and only if there exists choices c_1 to c_m some m some fixed number responses a_1 to a_k . So, choices guide the machine to make queries and obviously there will be answers which due to acceptance. So, but then we will in we are also encoding acceptance here and u_1 to u_k can be there are you into u_k satisfying assignments for the ins the cases where this is important when a_i equal to 1.

So, $\phi_i(u_i) = 1$ when a_i equal to 1 and such that x is a u_i . So, that is why it is all of these three things are coming under the existential quantifier. So, I will put a bracket and this thing for all we should be for all because if a_i is 0 then whatever input you try should be should not work. So, this \forall should be for all quantifiers. So, we have a there exists quantifier and we have a for all quantifier and such that n accepts x using choices c_1 to c_m and answers a_1 to a_k and.

So, this is the choices of this is acceptance of n and finally we also want to verify the correctness of the queries. So, this is just about n we also want to verify what the oracle did. So, that also we should be doing what is it that we should verify. So, whatever I had written here if a_i equal to 1 then it should we should verify that $\phi_i(u_i) = 1$. So, it is a, it is an implication that I have written here but as we have seen in previous lecture an implication can easily be converted to a Boolean formula.

And if a is 0 then $\phi_i \vee \psi_i$ should be 0 anyway the for the ψ_i there is an external universal quantifier. So, that will make sure that all the this applies for all the ψ_i 's and this we have to take an and over all the choices of i . So, in the in the 2 parts either the left side or the side may may get active depending on what a i 's. So, there is an existential quantifier followed by a universal quantifier and followed by this formula at the bottom and we have this part also.

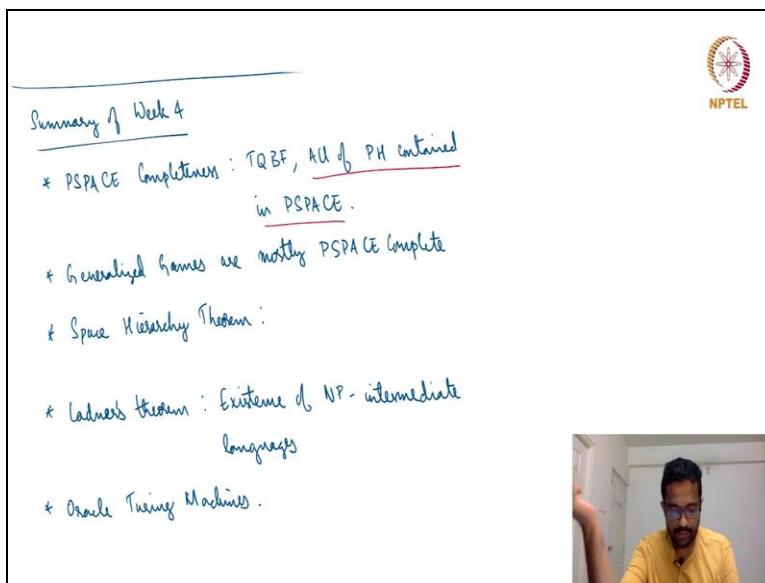
This light blue part but this light blue part is also it is just an non-deterministic it is just n accepts x using the choices m . So, all this is something that is verifiable in polynomial time. So, given m choices it is just a matter of verifying whether this non-deterministic choices lead to acceptance and what I have written the bottom this part also it is easy to verify by a deterministic polynomial time machine.

So, the entire thing is deterministic verifier with preceded by an existential and a universal quantifier. So, that is the proof that an arbitrary language in NP with with access to a SAT oracle is contained in Σ_2^P . So, basically we write down what it means for a language to be contained in NP with access to a SAT oracle. Once we write that down basically it is a bunch of non-deterministic choices with a bunch of queries and responses once we write that down we can encode it in a way with existential universal quantifier followed by a deterministic polynomial time verifier.

And that is about it that shows that NP with access to SAT oracle is contained in Σ_2^P and we have already seen the other direction. So, this shows that Σ_2^P is equal to NP with access to SAT oracle. And other things that I said earlier Σ_{i+1}^P is contained is equal to NP with access to q SAT i oracle and likewise for Π_{i+1}^P the proof follows very, very similarly. So, I will not get into that.

And one thing that you may wish to note here is that no I think that; no, so I was just trying to see if the quantifiers could be alternated but it is not possible here because the choices have to be made before v_1 to v_k . So, you cannot alternate the quantifiers and with that I have we have completed the description of how we can get the languages in polynomial hierarchy using the oracle turing machine description.

(Refer Slide Time: 23:20)



Summary of Week 4

- * PSPACE Completeness: TQBF, All of PH contained in PSPACE.
- * Generalized Games are mostly PSPACE Complete
- * Space Hierarchy Theorem:
- * Ladner's theorem: Existence of NP-intermediate languages
- * Oracle Turing Machines.

Now let me just summarize what we have seen in this week 4. We saw 4 or 5 different items the first thing that we saw was PSPACE completeness. So, we saw what is PSPACE what is PSPACE completeness, we saw the language TQBF and one point I felt I did not stress enough during the PSPACE lecture was that all of polynomial hierarchy is contained in PSPACE because the q SAT k which is the complete language for Σ_k can be viewed as an instance of TQBF.

Hence all of the complete languages of all the class all the classes in the polynomial hierarchy can all be reduced to the complete language or cannot be reduced to TQBF which is PSPACE complete. So, that is one point; PSPACE completeness and the fact that all of polynomial hierarchy is contained in PSPACE. We saw many instances of how games or the generalized instances of specific games such as chess are largely mostly PSPACE complete.

Because basically the game just boils down to analyzing a position drawing the game tree and at the end deciding a winner or loser is basically just a you can you can encode it in a Boolean formula. So, it just becomes like a fully quantified Boolean formula and hence it becomes a TQBF instance that is the reason why the generalized versions of games are PSPACE complete again the finite versions of games are all finite.

And then there is either a yes or no or whatever answer like tic-tac-toe. After this we saw Ladner's theorem which stated the existence of NP intermediate languages if P is not equal to NP. So, NP intermediate are the language R is a space between NP complete and polynomial time. So, it says that such a space exists if P and NP are different it cannot be that P is there NP complete is there and there is a void in between this the space in between is filled with languages.

And finally we saw oracle turing machines we saw the definition we saw the definition and we saw some we saw the turing reduction. And we saw how polynomial hierarchy the classes and polynomial hierarchy can be can be characterized as the as a result of the oracle turing machines. And one more point is that one point that I missed maybe I will just add in the middle sorry one point that I missed in the summary.

One point I missed in the summary which I will include now is space hierarchy theorem basically g is little o of f then $DSPACE$ of g is strictly contained in $DSPACE$ of f meaning there is something that you can compute with f space that you cannot compute with g space this is a space hierarchy theorem. This was proved also using diagonalization much like time hierarchy theorem.

So, this was the main concept and with space hierarchy theorem and PSPACE completeness space complexity space bounded complexity part is complete. Now we are venturing into other aspects lists such as loudness theorem and oracle turing machines. So, till space higher till space hierarchy theorem or still space complexity I was referring to Sipser now I do not think Sipser contained in many us I think it may contain some of these topics.

But I think it is better to refer to Arora Barak or Papadi Mithra's book for the for the remaining topics or you can also Google it there are many other lecture notes excellent lecture notes where people have exposed on this. Of course I will also share whatever I have been preparing but in addition to that you can also search for other material. And if you want other specific; there are specific request for other specific material please feel free to write to me in the forum I will be

happy to share. And with that I will conclude this lecture and conclude week 4 as well, thank you and see you next week.