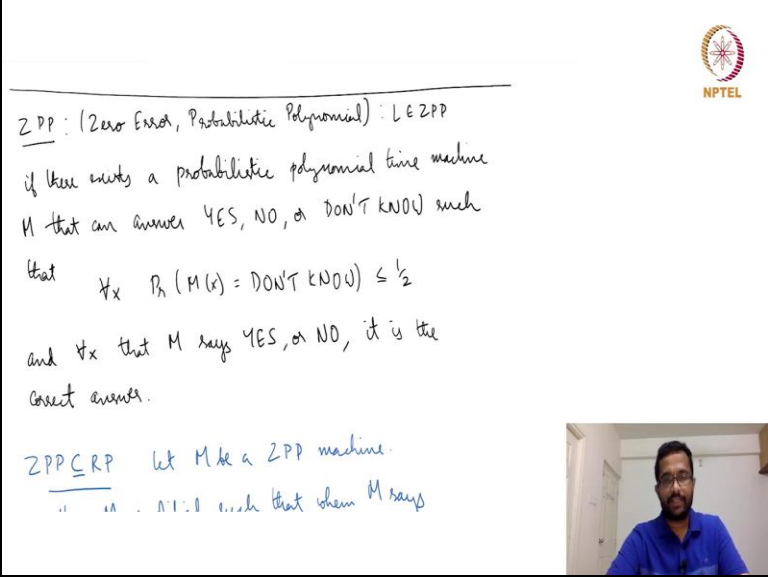



Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture -28
Randomized Complexity Classes Part 3

(Refer Slide Time: 00:15)





ZPP : (Zero Error, Probabilistic Polynomial) : LEZPP
if there exists a probabilistic polynomial time machine
M that can answer YES, NO, or DON'T KNOW such
that $\forall x \quad P_M(M(x) = \text{DON'T KNOW}) \leq \frac{1}{2}$
and $\forall x$ that M says YES, or NO, it is the
correct answer.

ZPP \subseteq RP let M be a ZPP machine.
.. .. n.t.o such that when M says

Hello and welcome to lecture 29 of the course computational complexity. So, in the past 2 lectures we have seen the complexity classes for randomized computation we have seen RP co-RP and BPP which are which correspond to the complexity class which correspond to Monte Carlo algorithms that run in a fixed some fixed time but then they could result in errors meaning. Yes yes instances could be reported as no or vice versa.

In this in this lecture I want to talk about ZPP which is a complicity class that pertains to Las Vegas algorithms where the running time varies but correctness is assured. Before getting into ZPP I want to state 1 small point which I think is an important point but I did not emphasize this in the previous lectures. The point that I want to make is that we have seen the probabilities throughout in the case of RP and co-RP and BPP we have seen these probabilities.

So, the way to understand these probabilities are that let us say consider this the BPP for instance the probability is always over the random choices made by the algorithm. So, when I say x is in

L implies the probability of acceptance at least 2-3rds means x is fixed and for any fixed x that is in L the probability of it getting accepted should be at least 2-3rds. And the probability is taken over the try random choices made by the made by the algorithm.

So, the algorithm may make many random choices and then it is taken over all the random choices possible for every fixed x that is in L. So, any fixed x there is a at least 2-3rds probability that it will be accepted and whenever x is not in L any fixed x that is not an L the probability of acceptance is at most 1-3rds. So, it is not like half the like 1x is there that will never be accepted not like that for any L any x that is in the language the probability is taken over the random choices or for that matter for any x whether it is in L or not in L the probability is taken over the random choices made by the algorithm for any fixed x.

(Refer Slide Time: 02:54)

Examples: ZPP.

ZPP: (Zero Error, Probabilistic Polynomial) : $L \in ZPP$

if there exists a probabilistic polynomial time machine M that can answer YES, NO, or DON'T KNOW such that

$$\forall x \quad P_M(M(x) = \text{DON'T KNOW}) \leq \frac{1}{2}$$

and $\forall x$ that M says YES, or NO, it is the correct answer.

So, let us go to ZPP, ZPP stands for zero error probabilistic polynomial. So, L is considered to be in ZPP if there is a probabilistic polynomial time machine M that can answer yes no or do not know. So, there is this is the new thing here is that we are allowing it and the machine to say I do not know such that the answer is always correct. So, you have a machine that is always correct that runs in polynomial time can use randomness but if you want the machine to be always correct you can always have the machine truthfully say I do not know.

So, this is not interesting like what uses a machine that always says I do not know. So, to make it interesting or to make it useful we limit the probability by which it can say I do not know for any x the probability for which it should say I do not know is upper bounded by half. So, with probability half it should give the correct answer. So, it cannot say yes and no both for the same input because it always has to output the correct answer.

So, if x is in the language it can either say yes or I do not know and the probability of saying I do not know is at most half if x is not in the language it could either say no or I do not know the probability of saying I do not know is at most half and it is not allowed to say an incorrect answer this is ZPP. So, when I make this definition it does not seem like what I just said the running time is the running time seems to be fixed in this definition.

But and there are no errors but then there is a provision of I do not know but we will later see in the same lecture we will see why or how we can change we can do some repetition tweaking etcetera. So, that the machine never says I do not know instead the running time becomes variable. So, by you can do a tradeoff between allowing I do not know and allowing it to take flexible time. We will see that later in this lecture.

So, now let us try to understand the classes at ZPP. So first thing to note is that ZPP is contained in RP why is that? Let us say there is a ZPP machine let us say M is FPP machine for a language L now to show that ZPP is contained in RP we should show that any language L is also in ZPP is also in RP. So, for we take the language L we take the ZPP machine for L that that is ZPP machine M . Now we will demonstrate an RP machine for L .

(Refer Slide Time: 06:03)

When $x \notin L$, then M always says no...
 $P_x [M(x) = \text{Acc}] = 0$

When $x \in L$, then M accepts w.p. $\geq \frac{1}{2}$
 $P_x [M(x) = \text{Acc}] \geq \frac{1}{2}$

So if $L \in \text{ZPP} \Rightarrow L \in \text{RP}$. That is $\text{ZPP} \subseteq \text{RP}$

Similarly $\text{ZPP} \subseteq \text{co-RP}$. $\text{ZPP} \subseteq \text{RP} \cap \text{co-RP}$ → EXERCISE

$\text{ZPP} = \text{RP} \cap \text{co-RP}$

11. $L = \{a, b, \dots\}$ then $\text{ZPP} \subseteq \text{RP} \cap \text{co-RP}$

Now we modify the machine it is a very simple trick. So, in an RP machine is not allowed to say I do not know it is allowed to say only yes and no. So, what do we do whenever M says I do not know the RP machine constructed. So, the RP machine we call it M prime M prime says reject. So, now M prime only says accept or reject yes or no. So, whenever x is not in the language M prime always says reject.

Because whenever x is not in the language M prime can say or M could have said I do not know or reject but now all the I do not knows have become reject also. So, whenever x is not in the language M prime always says reject. So, the probability of accepting such an x is 0 which is what RP wants us to do. Any x that is not in L property of acceptance should be at equal to zero. Whenever x is in L, M the ZPP machine could do 2 things it could accept or it could say I do not know.

The probability of saying I do not know is at most half but then now when we convert it to M prime all the I do not knows get converted to rejects. So, the probability of M accepting anyway the probability of M M saying I do not know was at least half sorry at most half. So, the probability of M accepting was at least half. So, the probability of M prime accepting is also at least half whenever M accepts n prime also accepts and prime only converts I do not know to reject.

So, again this is what RP machine RP wants. So, M prime so, by observing this you note that M prime is an RP machine for the language L . So, whenever an arbitrary language L is considered L from ZPP is considered you can show that L is in RP. You take the machine and whenever it says I do not know you make it in you make it say reject that is it. Similarly ZPP can be shown to be in co-RP this I would suggest you to work out as an exercise. This is exactly like the same thing you just have to change I do not know to accept here.

So, you can work out the details here I do not want to get into the details it is exactly like what we did here. So, ZPP is we saw that ZPP is an RP we also saw that ZPP is in co-RP. Again we did not see but I hope because of the extreme symmetry here you are convinced that ZPP is in co-RP but in any case you can convince yourself. So, consequently we can say that combining both of the things we can say that ZPP is a subset of RP and co-RP. So, it is a subset of the intersection.

(Refer Slide Time: 09:24)

M_2 (co-RP) decides for L .
 $x \in L \Rightarrow P_h(M_1(x) = \text{Acc}) \geq \frac{1}{2}, P_h(M_2(x) = \text{Acc}) = 1$
 $x \notin L \Rightarrow P_h(M_1(x) = \text{Acc}) = 0, P_h(M_2(x) = \text{Acc}) \leq \frac{1}{2}$
 M_1 accepts only when $x \in L$
 M_2 rejects only when $x \notin L$.
 M' : Run both M_1 and M_2 on x .
 Accept if M_1 accepts.
 Reject if M_2 rejects.

Now what is true is that it is equal to the intersection ZPP is equal to RP intersection co-RP. Let us see why? So, we have already seen that it is in RP we already seen that it is in co-RP. So, if it is in both its in the intersection now all that remains to be shown is the other direction inclusion we have to show that RP intersection co-RP is contained in ZPP we have to show that RP intersection co-RP is contained in ZPP. So, how do we do that? We take an arbitrary language in RP intersection co-RP and then show that this language is in ZPP.

So, let us consider an arbitrary language let us consider an arbitrary language L in $RP \cap co-RP$. So, it is in $RP \cap co-RP$ which means that it is an RP as well as within $co-RP$ which means it has an RP machine as well as a $co-RP$ machine that recognizes it. So, let M_1 be the RP machine for L and let M_2 be the $co-RP$ machine for L . So, now we know that again all I am doing is just writing down what we know so far.

So, many steps you will see are standard to show that something is a subset of the another thing you just take an arbitrary member of the first sub first set and then show that this arbitrary member belongs to the second set to show that 2 sets are equal you show $a \subset b$ and $b \subset a$ and this is all that I am using here. And once I took an arbitrary member of ZPP sorry we have to show that $RP \cap co-RP$ is in is contained in ZPP .

So, we took in our arbitrary member of $RP \cap co-RP$ and then what does it mean it is an RP . So, it has an RP decider it is in $co-RP$. So, it is a $co-RP$ decider. So, now let us write down what it means. So M_1 is an RP decider and M_2 is a $co-RP$ decider suppose x is in the language then the RP decider accepts with probability at least half and the $co-RP$ decider accepts with probability 1 this is what it does.

If x is not in the language RP decider accepts it probability equal to zero and $co-RP$ decider accepts it probability at most half. So, I am just writing down the definition of M_1 and M_2 based on what we saw for the definitions of RP and $co-RP$. So, what is what is to be noted here M_1 is a RP decider there is no false there is only false negative and it accepts there is no false positive. So, when M_1 accepts x is in the language when M_2 rejects x is certainly not in the language M_2 has only false positives there is no false negatives.

So, whenever M_2 rejects we know for sure that x is not in the language. Similarly whenever M_1 accepts we know for sure that x is in the language. So, let us try to understand how things are maybe again drawing a small grid picture.

(Refer Slide Time: 12:56)

NPTEL

M_2 rejects only when $x \notin L$.

M_1 Acc: $x \in L$
 M_1 Rej: $x \notin L$

M' : Run both M_1 and M_2 on x .
 Accept if M_1 accepts.
 Reject if M_2 rejects.
 Else say DON'T KNOW.

If $x \in L$, $P_x [M_1(x) \neq \text{Acc}] \leq \frac{1}{2}$
 So $P_x [M'(x) = \text{DON'T KNOW}] \leq \frac{1}{2}$
 Similarly, when $x \notin L$, $P_x [M'(x) = \text{DON'T KNOW}] \leq \frac{1}{2}$.

So, M_1 accept, I will write down what are the possibilities here M_1 accept M_2 accept and M_2 reject. So, can M_1 and M_2 both accept yes they can both accept and when M_1 and M_2 both accept we know that M_1 accepts only when x is in the language. So, when M_1 is accepting we know for sure that x is in the language even when M_2 is rejecting we know that x is in the language sorry this cannot happen because when M_1 accepts when M_2 rejects, M_2 so, this cannot happen sorry accepts because then certainly in the language.

So, M_2 cannot reject. So, this cannot happen suppose M_1 suppose M_2 rejects we certainly know that x is not in the language. So, M_1 also may reject and what can possibly happen is that M_1 rejects and M_2 accepts this can happen. So, 1 of the machines reject and the other machine accepts this certainly can happen and then what do we do. So, this is a confusing this is a situation that is that can happen. So, what we will do here.

So, we have. So, the problem here is to show that. So, 1 direction is already known ZPP is contained in RP decision we are showing that RP intersection co-RP is contained in ZPP. So, using the RP and co-RP decider we will build a ZPP decider. So, we need to build a ZPP decider. So, now the ZPP decider always has to be correct but it has an extra weapon it can say I do not know. So, when whenever M_1 accepts this situation that the top left quadrant it can say x is in the language x is in the language.

This situation does not arise stop write does not arise it cannot happen whenever M_2 rejects it one can say that x is not in the language. So, the only situation is this one when M_1 rejects and M_2 accepts which is possible because M_1 allows false negative and M_2 allows false positive. So, it could be that x is in the language and M_1 is reporting a false negative or x is not in the language and M_2 is reporting a false positive.

So, it could be either case but then a ZPP machine which is what we are trying to construct can always say I do not know. So, this is what we do here. So, let me just formally describe all that the M prime is the machine ZPP machine that we are going to construct this is a TZPP machine or let me say target ZPP machine. So, to build M prime we run both M_1 and M_2 on x . We accept if M_1 accepts and we reject if M_2 rejects.

So, as I said before it cannot happen that M_1 accepts and M_2 rejects and the confusing situation arises when M_1 rejects and M_2 accepts in that case we say I do not know. So, now to verify that this is a ZPP machine we need to do 2 things 1 is that the probability of I do not know is bounded to at most half on any input and 2 otherwise all the answers are always correct.

So, the second part is already done we know that M_1 accepts only when x is in L . We know that M_2 rejects only when x is not in L . So, we know for sure that the accept reject responses are correct and do not know responses are always correct. So, you can say I do not know any time and always be correct unless you know the answer but then machines do not do that.

(Refer Slide Time: 18:05)

So if $L \in ZPP \Rightarrow L \in RP$. That is $ZPP \subseteq RP$

Similarly $ZPP \subseteq co-RP$.

$ZPP \subseteq RP \cap co-RP$

$ZPP = RP \cap co-RP$

We have already seen $ZPP \subseteq RP \cap co-RP$

Let us consider $L \in RP \cap co-RP$. $\exists M_1 (RP)$ and $M_2 (co-RP)$ deciders for L .

$\forall x \Rightarrow P_1(M_1(x) = Acc) \geq \frac{1}{2}, P_1(M_2(x) = Acc) = 1$

So, the only thing that is left to be verified is that the probability of I do not know is at most half. So let us see suppose x is in the language how can it how can we say I do not know. If x is in the language then we say I do not know because M_1 rejected what is the probability that M_1 rejected well the probability that M_1 accepts when x is in the language is at least half. So, the probability that M_1 rejects is strict less than or equal to half.

So the probability that M_1 does not accept when x is in L is at most half and this is when M prime saying I do not know. So, the probability that M prime saying I do not know is at most half and prime says I do not know only when M_1 answers incorrectly and that happens when that happens with probability at most half. Similarly when x is not in L why does M prime say I do not know? This is happening because M_2 is accepting and what is the probability that M_2 accepts when x is not in L ?

The probability that M_2 is accepting when x is not in L is over here it is at most half and that is when we say M prime says I do not know. So, the probability of saying M prime saying I do not know is at most half. So therefore we can conclude that the machine M prime is a ZPP machine. So, whenever L is in RP and $co-RP$ it is in ZPP . Therefore we have shown that ZPP is equal to the class RP and $co-RP$. So, maybe it is an important result I will put it in a red box sorry. So, now finally 1 more point about ZPP this is what I promised at the beginning.

So, far we know ZPP as a as a type of machine that that runs in a fixed time but allows to say I do not know but I motivated ZPP by saying that it corresponds to Las Vegas algorithms one where the running time is variable but correctness is guaranteed. Now let us see how that that this definition corresponds to that.

(Refer Slide Time: 20:48)

Proof Sketch: Let $L \in ZPP$
 machine when it outputs DON'T KNOW till we get a YES/NO answer.
 * Say M is the ZPP machine.
 * M' : Run M . If M 's "DON'T KNOW", then Repeat.
 Repeat till we get a ACC/REJ.
 $E[\text{Running time of } M'] \leq \frac{1}{\frac{1}{2}} * \text{Running time of } M = 2 * \text{time}(M)$
 Other direction. Suppose there exists a machine ...
 to show statement Run the machine

So, this is the next theorem ZPP is the class of languages for which there is a probabilistic machine that runs in expected polynomial time. So, it is a probabilistic machine that runs in expected time $O(tn)$ where tn is a polynomial. So, expected polynomial time which means the running time can vary the expectation of running time is polynomial such that whenever M holds it correctly outputs the answer it may run for a long time that is but when it holds we get the correct answer and the expected running time should be polynomial.

So, in some particular input it may take a long time or in some in some cases may be just that all the randomness is not working out to our favor but it may take some time but the expected running time is polynomial. Again the expectation is calculated when the probability is taken over the random choices and not of array and not the choices of the random bits and not the choices of the input for any input this expectation should hold.

So, let us see why this is the case. So, there are 2 things here one is that we have already made a definition for ZPP that ZPP is a class of line class of languages that have a probabilistic

polynomial time machine that says yes, no and I do not know. But always answers correctly whenever it gives an answer. So, now from that definition we have to show that definition implies this definition or this definition this characterization and this characteristic implies that characterization.

So, we will I will give a brief proof sketch because again the proof details are simple. I will give a brief proof sketch suppose L is in ZPP which means it has as per the first definition it has a probabilistic polynomial time turing machine that says yes no and I do not know and probably if I do not know is at most half. What we can do is to repeat let us say so, now how do we build a machine that of this nature that I said here that the theorem statement says.

So, we need a machine that does not say I do not know but can take more it can take variable time. So, the simple way is sorry sorry again the simple way is to run the ZPP machine. So, maybe let us say M is a ZPP machine. So, M prime is what we will we will try to do M prime is a machine that will fit the characterization in this theorem statement. So, M prime is just this run M if M outputs do not know then repeat and repeat till sorry we get a accept reject or yes no.

So, we repeat it till we get accept or a reject. Let us see why this works what is the expected running time of M prime. So, certainly it will be repeated till it is it is accepting or rejecting and we know that M is a ZPP machine. So, M will always output the correct answer. So, that way it is all fine the only thing that remains to be verified here is that running time is an expected polynomial time. So, what is the running time of M prime or expected running time of M prime.

So, how many times will we have to repeat this? So the probability of so, whenever M says do not know it has to be repeated. So, how many times should we should it be repeated. So, that we get an accept or reject. So, does it remind you of some probability distribution that you know. So, if you have a die six sided die not normal six sided diet how many like what is the probability of getting a six it is 1 over 6.

Suppose you keep rolling till you get a 6 what is the number of expected number of rolls to get a 6 this is this corresponds to a geometric probability distribution and then expected running

expected number of trials is 1 divided by the probability of success. So, in the case of die it is it is six times. Here the probability of getting a correct answer is half or at least half. So, it is at least half. So, probabilities of saying do not know is at most half.

So, expected running time is at most 1 divided by probability of getting a accept reject is at least half. So, that is why we have to write at most multiplied by running time of M. And we know running time of M is polynomial time because that is how we define ZPP the running time is polynomial. So, this is equal to and this quantity is nothing but 2 2 into time taken for M. So, it is a polynomial time machine. So, the expected running time is polynomial or at most polynomial, so, which is fine.

So, whenever L is in ZPP we construct we get a machine as described in this theorem statement simply by running M again and again and again till it outputs a accept reject answer whenever it says I do not know you repeat and we know for sure that whenever it says accept or reject it is a correct answer.

(Refer Slide Time: 28:27)

The slide contains the following handwritten text:

- If M' does not complete, output "DO"
- EXERCISE: Complete the above proof with details of running time calculation
- Markov Inequality: If $x \geq 0$ always, then

$$P_n [x \geq a] \leq \frac{E[x]}{a}$$

$$P_n [x \geq 2E(x)] \leq \frac{1}{2}$$

The slide also features the NPTEL logo in the top right corner and a small video inset in the bottom right corner showing a man in a blue shirt speaking.

Now the other direction; suppose there is a machine like this which runs in expected polynomial time. So, the running time can vary but at the end whenever it halts it will give the correct answer now how do we go to a machine which has a fixed running time? That is also not very

difficult. Suppose there is a machine as we said in the theorem statement which means it has running time that is expected running time is something.

Now the idea is to run the machine for a certain time. So, certain time what works is. So, let us say the expected running time is maybe this works I think 2 times the expected time certain. So, this works I think and the thing is it may not conclude because the statement the theorem statement we say it may run the bound is known only for the expected running time a specific a specific input may take long time depending on the random choices made.

But we know that the expectation is fixed. So, we can and we can run allow it to run till certain time if it does not complete its execution till that certain time if it does not complete its execution until that certain time we just stop the execution and say I do not know. Because now in the ZPP model we cannot have the flexibility of running forever but we do have the flexibility of saying I do not know. So if it if it has not completed the execution you say I do not know. So, you run a machine M .

So, maybe I will just write in a bit more detail write in a bit more detail say M is the machine as in the theorem statement. So, the ZPP machine will be M prime sorry or I will call M prime the machine in the theorem statement just because earlier also n prime corresponding to the theorem statement and ZPP machine will be M . So, what is M start M prime run till time expected time of M prime sorry run till time 2 multiplied by expected time of M prime 2 multiplied by this if M prime completes output the same as M prime.

If M prime does not complete then we are stopping it does not complete output do not know. So this works because of what we have here called Markov's inequality when a random variable is non-negative when a random variable is non-negative we have the probability that it exceeds a certain quantity is at most expectation of the random variable divided by a . So, where a is the quantity that we are doing this in other words sorry.

So, when we the probability that it is more than 2 times expectation is at most half. So, when you apply this for x equal to the running time sorry for the expected the running time f prime it works

out to be the probability that you end up saying do not know is the probability that you exceed the running time of the expectation of twice the expectation. So, the probability that you exceed x exceeds twice his expectation is at most half and that is the probability by which you will see I do not know.

And correctness it runs in polynomial time because you are stopping the machine at a certain time. So, you know that it runs at a certain time and the answers are always correct. As an exercise you could just I just I gave most of the details but if you want to convince yourself you can work out the rest of the details. And with that I think I will conclude this lecture and I will quickly summarize we stated the model ZPP or the complexity class ZPP for which we saw we gave 2 definitions.

One is the Las Vegas algorithms where which we saw at the end where the machine will always be correct but the running time is variable. And the second definition which we saw first was that the running time is fixed but the machine can do 2 or 3 things yes no or do not know. So, it has a do not know option but the probability for by which it can output do not know is limited. And the thing that we observed is that ZPP is equal to the intersection of RP and co-RP.

And the interrelations between all these are very interesting like you see how we proved ZPP is equal to RP intersection co-RP and also the proof that the proof that both the correct both the definitions that ZPP is the model with yes no and do not know and the model where the running time is running time is variable but the correctness is guaranteed with only yes and no and both of these use make use of interesting probability and some and simple but interesting tricks.

We will conclude with the summary of week 5 we first saw the baker gill solid theorem which said that techniques that relativize in particular diagonalization cannot be used to resolve the P versus NP question. So, this was done by demonstrating 2 oracles a and b such that P to the A is equal to NP to the A and P to the B is not equal to NP to the B . So because of this these 2 oracles we cannot use techniques that relate device to show or to resolve P versus NP.

After the proof of the Baker Gill Solovay theorem we went to the randomized complexity classes starting with the motivation then we first saw the classes RP, co-RP and BPP which were classes that correspond to Monte Carlo algorithms. We saw how to do boosting for each of these classes how we could repeat the algorithm and get improve the error errors. We saw how these complexity classes relate to each other we also saw ZPP complicity class pertaining to Las Vegas algorithms.

We also saw how this class relates to the other classes that I have already mentioned. We saw multiple we saw at least 2 characterizations of ZPP and how the equivalence between them we also saw the proof that ZPP is equal to RP intersection co-RP and with that I think we concluded this week. And next week we will we will start off where we where we are from this point we will see some more randomized complexity classes. How they relate to some other things that we have already seen and take it forward from there.