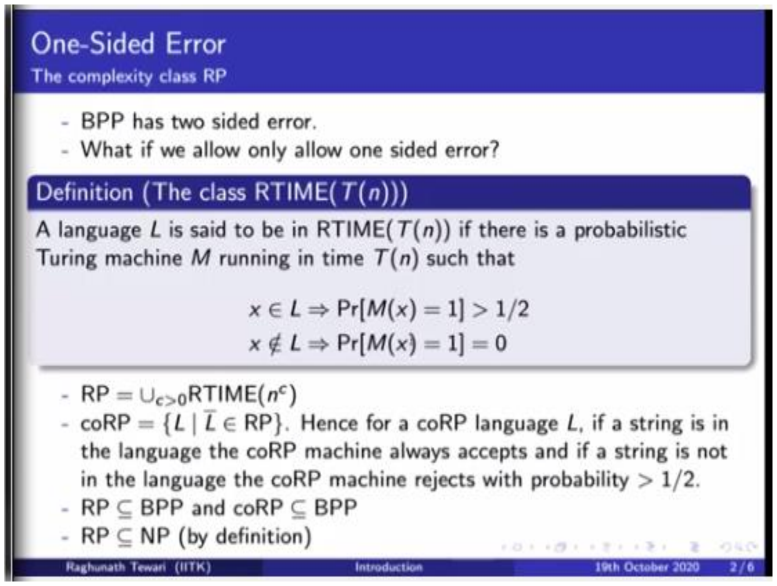
Computation Complexity Theory
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture-20
One-sided and Zero-sided Error Probabilistic Complexity Classes

Hello everyone, so we will continue our discussion of probabilistic turing machines. Today we look at probabilistic turing machines which have one sided and zero sided error. So I am going to start by mentioning what we mean by one sided and zero sided error machines, and how do they compare with whatever we have seen about probabilistic turing machines so far.

(Refer Slide Time: 00:40)



One-Sided Error
The complexity class RP

- BPP has two sided error.
- What if we allow only allow one sided error?

Definition (The class RTIME($T(n)$))
A language L is said to be in RTIME($T(n)$) if there is a probabilistic Turing machine M running in time $T(n)$ such that

$$x \in L \Rightarrow \Pr[M(x) = 1] > 1/2$$
$$x \notin L \Rightarrow \Pr[M(x) = 1] = 0$$

- RP $= \cup_{c>0}$RTIME(n^c)
- coRP $= \{L \mid \overline{L} \in \text{RP}\}$. Hence for a coRP language L, if a string is in the language the coRP machine always accepts and if a string is not in the language the coRP machine rejects with probability $> 1/2$.
- RP $\subseteq$ BPP and coRP $\subseteq$ BPP
- RP $\subseteq$ NP (by definition)

Raghunath Tewari (IITK) Introduction 19th October 2020 2 / 6

So, the class BPP that we saw had two sided error. So, what do we mean by two sided error? So, what it means is that if the string x belongs to the language, even then there is a probability that the machine might give a wrong answer, the machine might reject. And if does not belong to the language, then also there is a probability that the machine might accept. So, in both cases, there is a small probability that the machine can make a error. So, this is what is known as two sided error.

Now what if we allow only one sided error? What if we do not allow any error to happen on either when x is in L or x is not in L, what kind of complexity class do we get, and how do they compare with the two sided error classes? So, we define the class RTIME T of n analogous to D time T of n as follows. So, a language L is said to be in RTIME T of n, if there is a probabilistic during machine M running in time T of n.

Such that if x belongs to L, then the probability that the machine accepts is greater than half. And if x does not belong to L, then the probability that the machine accepts is 0 or in other words the probability that the machine rejects is 1. Now if you compare this definition to the definition of BP TIME T of n, you would realize that in BP times T of n, there is a probability of an error in both sides, when x is in L and when x is not in L.

Here when x is not in L, we do not allow any error to occur. So, when x is not in L the machine always gives the correct answer. Now the class that we get by looking at RTIME T of n when T of n is polynomial is the class RP. So, this is analogous to the class BPP for one sided error machines. And now we can define the complement of this class, we can define the class coRP as the set of all languages L, such that the complement of L belongs to RP.

So, what this means is that if we have a language which is in coRP string, if a string is in the language L, then the coRP machine will always accept that string. And if a string is not in the language then the coRP machine will reject with probability greater than half. So, one small point that I want to make before we proceed forward is that although here we choose the number half as a constant.

One can actually arbitrarily increase this probability by a significant amount or by let us say 1 - 1 over a polynomial. So, I can always by repeating the same computation, I mean by basically running the same input on the machine again and again. And by taking, so if the machine is let us say, suppose I take an RTIME Machine, which has these parameters, that is if x is in L, then it is probability of accepting is greater than half and if x is not in L, then the probability of accepting is 0.

So, if I run this machine again and again and whenever I have a yes answer, I just halt an output yes, whenever I have a no answer, I keep on repeating this experiment. So, if we actually repeat this experiment, a sufficient number of times, one can show that the probability can be brought very close to 1 the probability of correctness. Or in other words, the probability of the machine making an error will be very small.

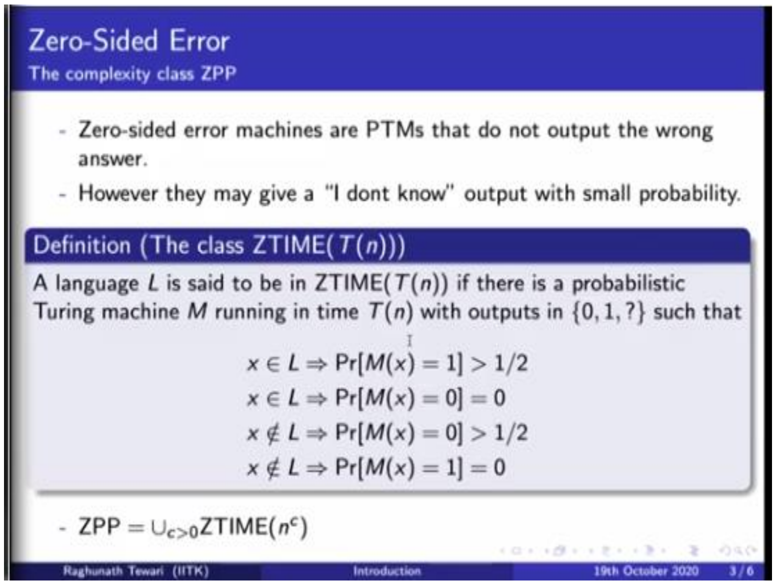
So, this actually we will be discussing in our next lecture more thoroughly when we talk about strong BPP and weak BPP and how we can use (()) (05:16) to improve their I am show that these classes are equal. So, coming back to the class RP or coming back to the base definition of RP. Next is that one can easily show that RP is contained in BPP and coRP is also contained in BPP, so this actually follows directly from the definition.

So, in BPP, although here we have just greater than half in BPP we require it to be greater than two third. But as I said that by this running this a few times one can increase the probability of correctness to greater than any constant**.** And the last relation that I would like to state is that RP is contained in NP is contained in the class NP. And this also actually follows by definition, because just observe the following that if x is not in L.

So, think of a probabilistic turing machine as a turing machine where it is a non deterministic turing machine. And so, whenever the probabilistic turing machine does a coin toss to decide which direction it wants to go to. The non deterministic turing machine also basically does the same thing, it non deterministically chooses which direction it wants to go to. Now if x is not in L, then observe that no computation part of this machine accepts which is exactly what a non deterministic machine should be.

On the other hand, if x is in L, here what we have is that at least half the many computation parts accept. In other words at least one part definitely accepts it is much more than 1. So, therefore a probabilistic RP machine by definition is also an NP machine because if the input is in the language, we have at least one accepting part**.**

(Refer Slide Time: 07:25)



Zero-Sided Error

The complexity class ZPP

- Zero-sided error machines are PTMs that do not output the wrong answer.
- However they may give a "I dont know" output with small probability.

Definition (The class ZTIME($T(n)$))

A language L is said to be in ZTIME($T(n)$) if there is a probabilistic Turing machine M running in time $T(n)$ with outputs in $\{0, 1, ?\}$ such that

$$x \in L \Rightarrow \Pr[M(x) = 1] > 1/2$$
$$x \in L \Rightarrow \Pr[M(x) = 0] = 0$$
$$x \notin L \Rightarrow \Pr[M(x) = 0] > 1/2$$
$$x \notin L \Rightarrow \Pr[M(x) = 1] = 0$$

- ZPP $= \cup_{c>0}$ZTIME(n^c)

Raghunath Tewari (IITK) Introduction 19th October 2020 3 / 6

Next we move to zero sided error. So in BPP, we had errors in both sides, in RP we had error only in one side, RP and coRP both. Now what do we mean by zero sided error? So, zero sided error machines are probabilistic turing machines that do not output the wrong answer ever. So, if it does not output the wrong answer, then what do we mean by probabilistic? How is it a probabilistic?

So here, what we allow is that the machine might output a I do not know answer with a small probability or a question mark output with a small probability. What do we mean by that? So, formally, if I want to look at it, so a language L is said to be in the class ZTIME T of n, if there is a probabilistic during machine M running in time T of M with outputs from the following set 0, 1 and ?, it will give three types of output.

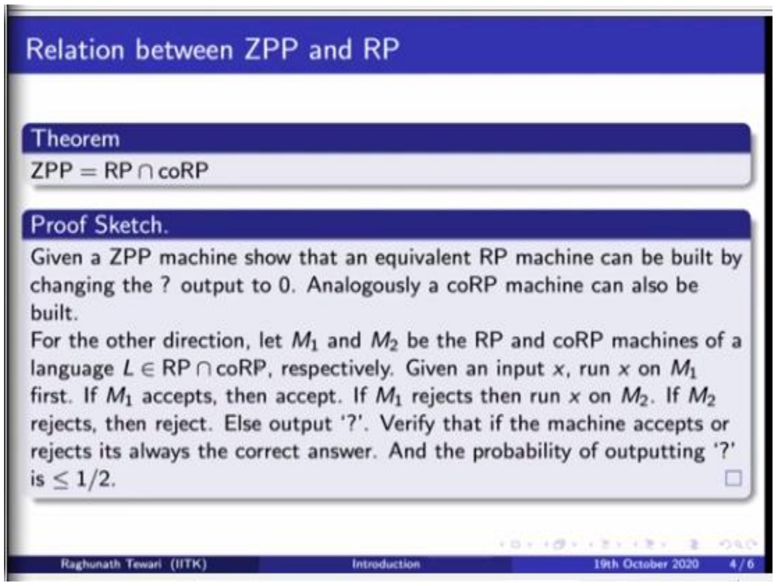
Now if x belongs to the language, then the probability that the machine accepts is greater than half and the probability that the machine rejects is 0, a machine will never reject. This is what we mean by saying that the machine will not give a wrong answer. So, what it does in the remaining computation paths is that it just outputs ?. So, in at least half the fraction of the computation paths, it outputs 1 and or it accepts.

And in the remaining computation paths, it just output ? or I do not know. Similarly if x is not in L, then the probability that the machine rejects is greater than half and the probability that the machine accepts is 0. So, once again, if x is not in L, then on more than half the fraction of the computation paths the machine would reject and on the remaining computation part it again outputs I do not know.

So, it will never give up. If x is in L, it never rejects and if x is not in L it would never accept. So, the polynomial time analogue class of ZTIME T of n is known as ZPP. Now what is the relation between these classes BPP, RP and ZPP. So, we have already seen that RP and coRP both are contained in BPP by just the definition. And if you look at the class ZPP, again observe that ZPP is actually contained in the class RP.

How do we show that it is contained in RP? So, we just have to make one modification, so I will just come to that in a moment. If that one can show that ZPP is contained both in RP and coRP.

(Refer Slide Time: 10:41)



Relation between ZPP and RP

Theorem

ZPP $=$ RP $\cap$ coRP

Proof Sketch.

Given a ZPP machine show that an equivalent RP machine can be built by changing the ? output to 0. Analogously a coRP machine can also be built.

For the other direction, let M_1 and M_2 be the RP and coRP machines of a language $L \in$ RP $\cap$ coRP, respectively. Given an input x, run x on M_1 first. If M_1 accepts, then accept. If M_1 rejects then run x on M_2. If M_2 rejects, then reject. Else output '?'. Verify that if the machine accepts or rejects its always the correct answer. And the probability of outputting '?' is $\leq 1/2$. $\square$

Raghunath Tewari (IITK) Introduction 19th October 2020 4 / 6

So, relation between ZPP and RP, so what one can show is that not only is ZPP contained in both RP and coRP, actually ZPP is exactly equal to the intersection of RP and coRP. And it is easy result, but nevertheless it is interesting result. So, how do we prove this? I will just give a proof sketch. So, first we showed that ZPP is a subset of RP intersection coRP. So, given a ZPP machine, we show an equivalent RP machine for that machine**.**

And how do we build that RP machine? So, we build that RP machine by changing the ? output to a reject output or 0 output. Now what happens in this case, just analyze? So, if the ZPP machine was accepting, so suppose if I had an input, which was in the language. Now we know that the ZPP machine accepts with probability greater than half. And the RP machine also now accepts with probability greater than half.

Because we are not changing the accepting criteria over here. On the other hand, if we had an input x which was not in the language, then the ZPP machine was rejecting it with probability half and it was going to the question mark output with probability, the remaining probability. And now what happens is that, because I am changing ? to 0 the ZPP machine will always be rejecting if x is not in L.

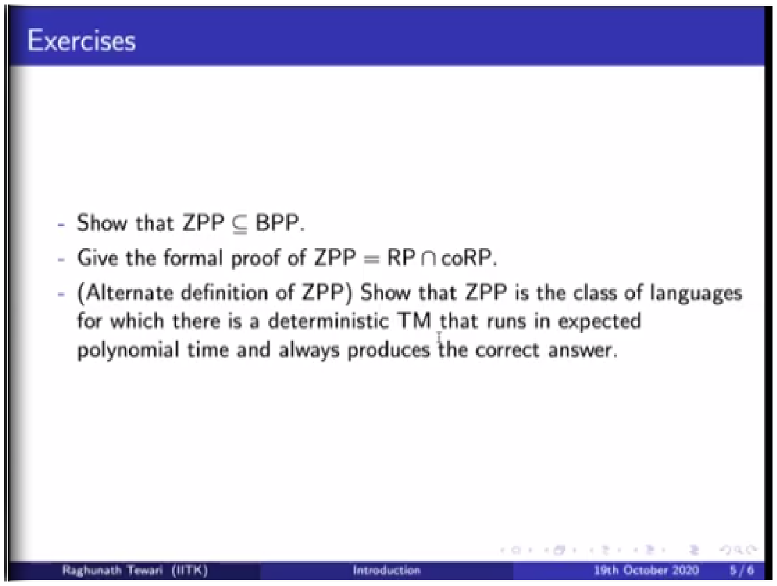
So, therefore this shows that the ZPP machine can be converted to an RP machine by this kind of a modification. And analogously you can also convert a ZPP machine to a coRP machine. So, this proves that if a language L is in ZPP, then it also belongs to RP as well as coRP. So it is in RP intersection coRP. Now what about the converse direction? So, the converse direction is suppose if I have a language, which is in RP intersection coRP, how do we prove that it is in ZPP?

So, let us say that a language is in RP intersection coRP via the machines M 1 and M 2 respectively, so M 1 is a RP machine for L and M 2 is a coRP machine for L. Now given an input x, how do we build the ZPP machine? So, given an input x, first what we do is, we run x on the machine M 1, if M 1 accepts then we just halt and accept. If M 1 rejects, then we run x on the machine M 2, if M 2 rejects we again halt and reject otherwise we output ?.

Now verify that if x was indeed in the language, then the machine M 1 always accept with probability greater than half, therefore your entire construction also accepts with probability greater than half. On the other hand, if and it is access a probability greater than half and it would never actually reject. Because if x is in the language, M 1 might reject.

So, if M 1 is rejecting your anyway going to M 2, but M 2 would never be rejecting, because M 2 is a coRP machine, M 2 would always accept if x belongs to the language. So therefore, so M 2 always accept if x belongs to the language. So, therefore the machine accepts with probability greater than half and for the remaining probability it would output ?. And similarly if x was not in the language analogously one can show, that again the machine will give the correct answer. So, I will just leave this sketch, I will just leave the complete proof to you as an exercise.

(Refer Slide Time: 15:23)



Exercises

- Show that ZPP $\subseteq$ BPP.
- Give the formal proof of ZPP $=$ RP $\cap$ coRP.
- (Alternate definition of ZPP) Show that ZPP is the class of languages for which there is a deterministic TM that runs in expected polynomial time and always produces the correct answer.

Raghunath Tewari (IITK) Introduction 19th October 2020 5 / 6

In fact I have the following exercises. So, firstly show that ZPP is contained in BPP or, so how does one prove this? Next exercise is give a formal proof of the result that I just mentioned. So, the one in the previous slide is just as proof sketch you have to actually modify it a little formally. And then you have to argue that ZPPS is RP intersection coRP.

Of courses you also have to argue that whatever construction you are giving it runs in polynomial time. The third exercise in fact, the most interesting exercise is to show this alternate definition of ZPP. So, one can analogous one can alternatively define ZPP as the class of languages for which there is a deterministic turing machine that runs in expected polynomial time and always produces the correct answer.

So, what this means is that, for some inputs the machine can take a longer time or longer than polynomial time, it can actually take maybe even exponential time. But on an average the turing machine runs in expected time. Basically if I take an average over all inputs, then the running time of the turing machine is polynomial. So, this proved that this is a alternate definition of ZPP.

In other words if you take a language that is in ZPP by the previous definition, then that language is also in this class, that I have defined in this third exercise. And if you take a language in this class, then it is also in ZPP, both directions you need to show. So, that is all that I had for this lecture, thank you.