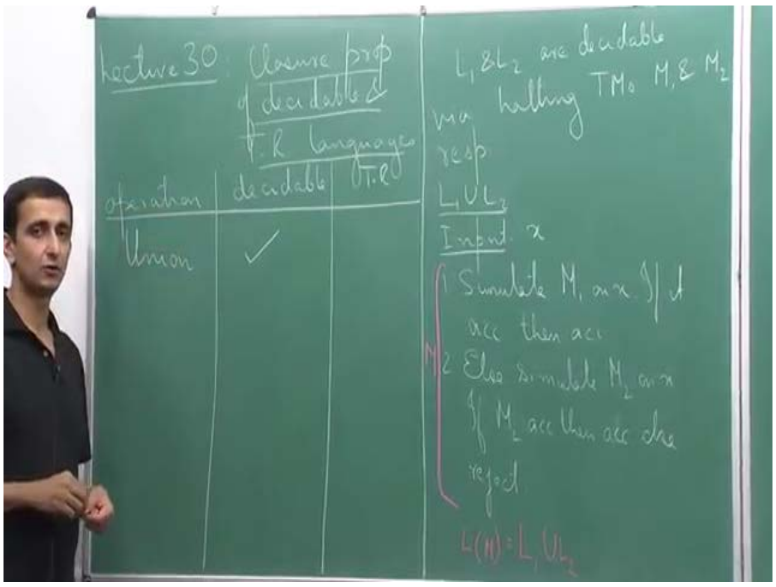
Theory of Computation

Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 30
Closure Properties of Decidable and Turing recognizable languages

Welcome to lecture 30 of this course. Today, we are going to study Closure Properties of Decidable languages and Turing recognizable languages,

(Refer Slide Time: 00:51)



Lecture 30: Closure prop of decidable & T.R languages

operation	decidable	T.R
Union	✓	

L_1 & L_2 are decidable via halting TMs M_1 & M_2 resp

$L_1 \cup L_2$

Input: x

M
1. Simulate M_1 on x. If it acc then acc
2. Else simulate M_2 on x If M_2 acc then acc else reject

$L(M) = L_1 \cup L_2$

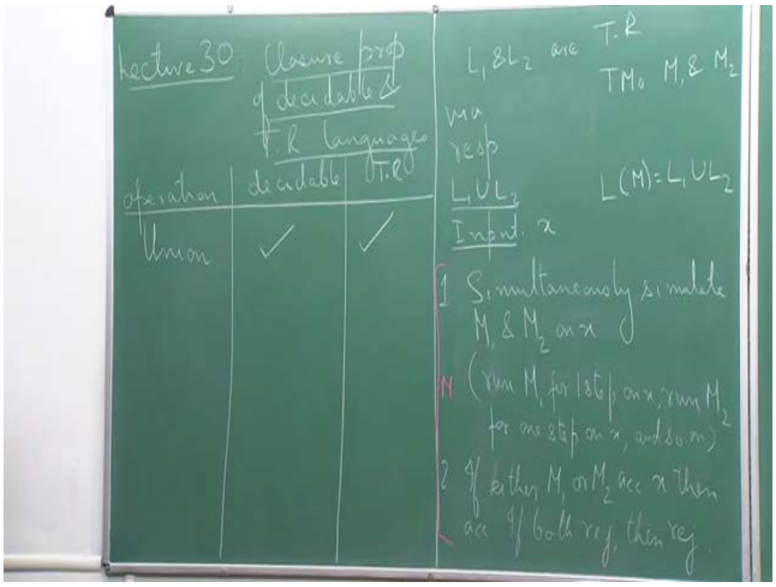
So, under some of the standard operations such as union, concatenation, star, intersection and compliment, let us create a summary table of the operations and the corresponding class of languages and then we will see the proof of which have these classes closed under which operations. Let me create a table. So, we have operation when we have decidable languages and let us say we have Turing recognizable languages. So, T R stands for Turing recognizable.

The first operation that we will see is union. So, our decidable languages and Turing recognizable languages closed under union. Let us try to see. Let us take two languages L 1 and L 2, which are decidable. So, L 1 and L 2 are decidable via halting Turing machines M 1 and let say M 2 respectively. So, what can we say about L 1 union L 2 is L 1 and union L 2 also decidable. The answer is yes, L 1 union L 2 is also decidable. And to prove this observes that what we can do is that. So, we can first given an input so given x, so let say we have input x.

So, what we can first do is simulate M 1 on x, if it accepts then accept, else simulate M 2 on x; if M 2 accepts then accept, else reject. So, because both M 1 and M 2 are halting Turing machine the design that we are giving here. So, this is let say design of a machine, let us call it M. So, this is let us say - Turing machine M. So, what will happen is that by this argument the language of M would be equal to the language L 1 union L 2, because M 1 and M 2 are halting Turing machines, M will always halt. If x belongs to either the language of M 1 or the language of M 2, then M would accept it; otherwise, it would reject. So, this proves that decidable languages are closed under union.

Now, what about Turing recognizable languages, so can we do the same thing for Turing recognizable. So, suppose if x belong to the language of M 1, then we were fine, but consider the following case what if x is in L 2 that is x belongs to the language of M 2 and on M 1 x loops forever it never stops. What happens, then if we actually try to do this kind of a simulation they run into a problem, because when I simulate M 1 on x it goes on forever, it never stops, all though x actually belongs to L 1 union L 2 because it belongs to the language L 2. So, actually we are never able to accept x using this machine M, if you do this kind of simulation. So, what is the idea? How can we modify the construction of the machine M, so that it would accept every string that belongs to either L 1 or L 2?

(Refer Slide Time: 05:39)



Lecture 30 Closure prop of decidable & T.R languages

operation	decidable	T.R
Union	✓	✓

L_1 & L_2 are T.R

TMs M_1 & M_2

via resp

$L_1 \cup L_2$

$L(M) = L_1 \cup L_2$

Input: x

M
1 Simultaneously simulate M_1 & M_2 on x

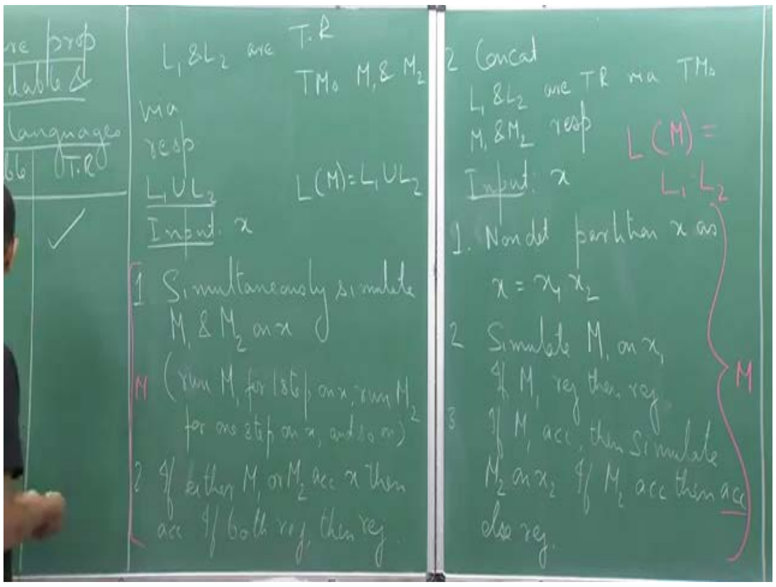
(run M_1 for 1 step on x; run M_2 for one step on x, and so on)

2 If either M_1 or M_2 acc x then acc. If both rej, then rej.

So, what we do is we modify the machine M in the following manner. So, given input x, what we will do is we will simultaneously run M 1 and M 2 on the input x. In other words, we will run M 1 for one step on x, then we will run M 2 for one step, then we again run M 2 for one step then again M 2 for another step and so on. We do not allow one machine to take control of x at a time. So, it is just that the simultaneously run the input x and if one of the machines ever accept, then we accept. If none of the machines accept for both the machines go and infinite loop then of course, we will go into an infinite loop, but it does not matter because then what it implies is that x does not belong to L 1 union L 2.

So, what we do is we simultaneously simulate M 1 and M 2 on x. In other words, if I write it in bracket, so run M 1 for one step on x, and then run M 2 for one step on x and so on. If either M 1 or M 2 accepts x, then accept; if both reject then reject. And if both of them go into an infinite loop; or, if one rejects, and the other goes in to an infinite loop even then we do not do anything, we just go into an infinite, does not matter. So, in this manner, basically if we do this then what we can say is that if L 1 and L 2 instead of being decidable, if they are Turing recognizable by halting machines M 1 and M 2 respectively then this construction of machine M ensure that L of M is equal to L 1 union L 2. So, this proves that Turing recognizable languages are also closed under union.

(Refer Slide Time: 09:16)



L_1 & L_2 are T.R via TMs M_1 & M_2 resp

$L_1 \cup L_2$ $L(M) = L_1 \cup L_2$

Input: x

1 Simultaneously simulate M_1 & M_2 on x

M (run M_1 for 1 step on x, run M_2 for one step on x, and so on)

2 If either M_1 or M_2 acc x then acc If both rej, then rej

2 Concat

L_1 & L_2 are TR via TMs M_1 & M_2 resp

$L(M) = L_1 L_2$

Input: x

1. Nondet partition x as $x = x_1 x_2$

2 Simulate M_1 on x_1 If M_1 rej then rej

3 If M_1 acc, then Simulate M_2 on x_2 If M_2 acc then acc else rej.

M

Now, let us look at the second property. The second property will be the concatenation operation. So, suppose L 1 and L 2. Now, what I will do is that for concatenation, I will only prove it for Turing recognizable languages. The case of decidable languages is actually similar. So, suppose L 1 and L 2 is Turing recognizable via Turing machines M and M 2 respectively.

Then what we have is that we have that L 1 is L of M 1 and L 2 is L of M 2. Now given input x, I want to design a machine, which would accept x if x a concatenation of two strings one belonging to L 1, and the other belonging to L 2. The idea is how do I guess this concatenation, how do I guess this dividing point of the string x, where x get divided into two parts let say x 1 and x 2, such that x 1 belongs to L 1 and x 2 belongs to L 2.

So, to do this, what we do is that we will use non determinism I mean of course, we can do this deterministically also there are other algorithms also that one can design, but I a this going to give application of non-determinism here. So, non-deterministically partition x as x equals x 1 x 2. So, what do we mean by doing this non-deterministically, how do we write a nondeterministic algorithm that does this.

So, what the nondeterministic algorithm will essentially do is that, if you look at the length of x lets say the length of x is may be 85, what it will do is that it will just guess a number between 0 and 85. By guessing a number, it just basically guesses a sequence of binary symbols such that the symbols basically give a number that is not more than 85; if it is more than 85, we can always discard.

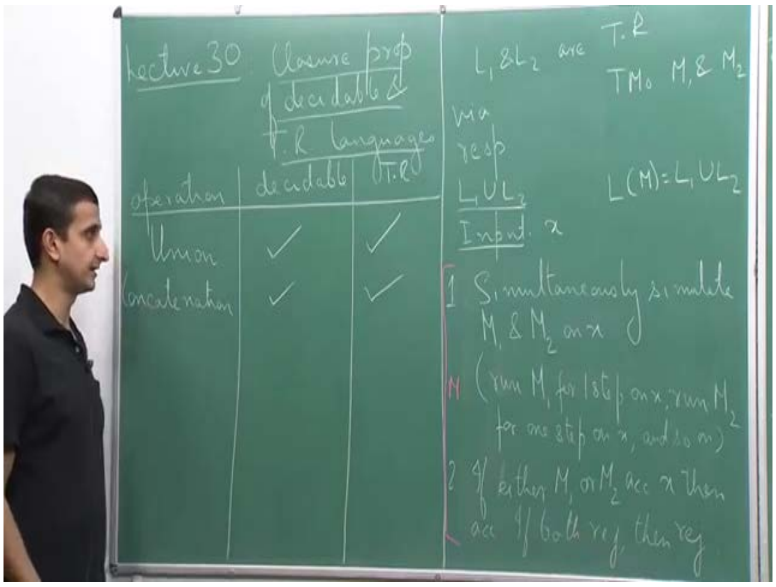
It will guess a number between 0 and the length of the x whatever it is. And then what it will do is that it will set x 1 to be the string which is the first that many number of bits. Let us say the guest number is L. So, if we look at the first L bits of x and it will set x 1 to be L, it will set x 1 to be the string come consisting of the first L bits of x and it will set x 2 to be the reminder part of the string. Now, once this is done, you simulate M 1 on x 1. If M 1 rejects, then reject; if M 1 accepts then simulate M 2 on x 2. If M 2 accepts then accept, else reject.

Let us analyze this algorithm, when does this algorithm accept x. So, observe that this algorithm will accept x only in this step. So, only in this particular point, it is accepting; no other place it is accepting. So, when does it accept? So, it is accepting first of all if M 1 accepts the string x 1 and secondly, if M 2 accepts that string x 2, so that means, if x 1 belongs to L 1 and if x 2 belongs to L 2 that is the only case in which it is accepting so that is one side. Now, what if it is not, so on the other side, if M 1 rejects x 1, then anywhere we have directly rejecting, we are not even going into the third step. Or if M 2 is rejecting x 2, even then we are rejecting so that means, if one of these two does not belong to L 1 or L 2, even then we are rejecting.

Thirds case can occur if let say one of them goes into an infinite loop on the corresponding string that is M 1 goes into an infinite loop on x 1, even then we do not reach the third step which means we actually do not accept x. Or if M 2 goes into an infinite loop on x 2 even then we do not accept which means that x is not accepted. The only case in which x is accepted is if there exist a point say if that can be divided into two parts, there both belongs to the corresponding languages.

And in a similar manner, you can show if L 1 and L 2 are decidable then L 1 concatenated with L 2 is decidable. So, what this shows is that L of the machine M, where M is this machine that we have designed. So, L of M is equal to L 1 concatenated with L 2. So, this shows that if I look at the concatenation operation decidable and Turing recognizable languages are closed under it.

(Refer Slide Time: 16:35)



Lecture 30

Closure prop of decidable & T.R languages

operation	decidable	T.R
Union	✓	✓
Concatenation	✓	✓

L_1 & L_2 are T.R via TMs M_1 & M_2 resp

$L_1 \cup L_2$

$L(M) = L_1 \cup L_2$

Input: x

1 Simultaneously simulate M_1 & M_2 on x

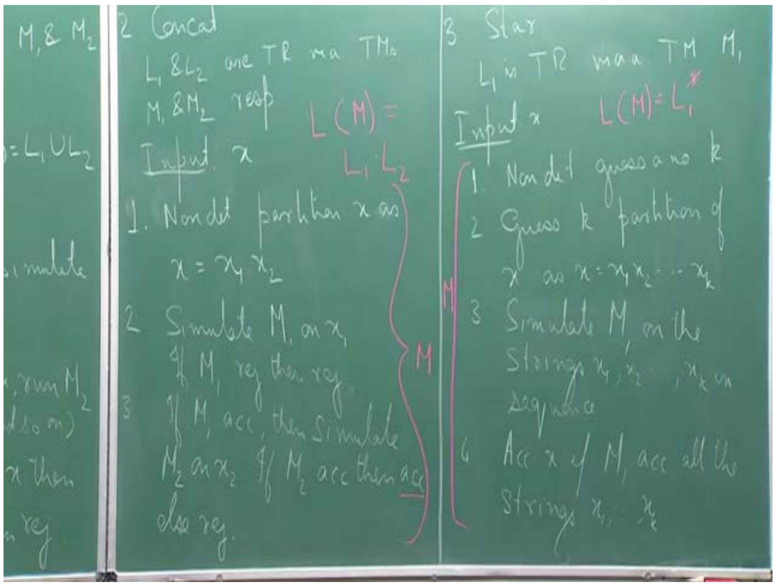
(run M_1 for 1 step on x; run M_2 for one step on x, and so on)

2 If either M_1 or M_2 acc x then acc If both rej, then rej

M

Now, let us move on to the third operation, that is star.

(Refer Slide Time: 16:48)



2 Concat

L_1 & L_2 are TR via TMs M_1 & M_2 resp

$L(M) = L_1 L_2$

Input: x

1. Non det partition x as $x = x_1 x_2$

2 Simulate M_1 on x_1 If M_1 rej then rej

3 If M_1 acc, then Simulate M_2 on x_2 If M_2 acc then acc else rej.

M

3 Star

L_1 is TR via a TM M_1

Input x $L(M) = L_1^*$

1. Non det guess a no k

2 Guess k partition of x as $x = x_1 x_2 \cdots x_k$

3 Simulate M_1 on the Strings $x_1, x_2, \ldots, x_k$ in sequence

6 Acc x if M_1 acc all the Strings $x_1, \ldots x_k$

M

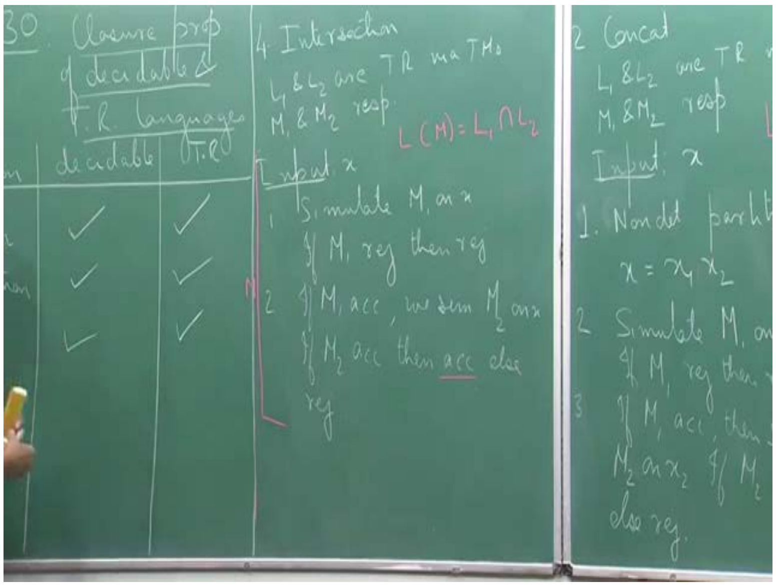
The star operation is also very similar to concatenation; it is basically concatenation of multiple strings from the same language. So, suppose if L 1 is Turing recognizable then via a Turing machine M 1, what we do is that given input x, we first we non-deterministically guess a number k, guess k partition of x as x equals x 1 x 2 up to x k. So, this is the same. So, we just have one additional step here, where we guess a number k and then we choose basically we do the same as concatenation. But instead of just guessing a partition consisting of two strings, we guess a partition consisting of k strings.

Now accept x if and the let me not if and only I will accept x. So let me (Refer Time: 18:52) sentence here. So, simulate M 1, so this machine M 1 on the strings x 1, x 2 up to x k in sequence. And accept x, if M 1 accepts all the strings x 1 through x k, so that is the only case where you accept. So, once again, this is the design of the machine M and what this gives is that L of M in this case is L 1 star. So, this shows that under the star operation also decidable and Turing recognizable language is closed.

(Refer Slide Time: 20:15)

So, once again all though for the star operation, we showed the case of only Turing recognizable language, but the case of decidable language is actually similar it is not any different.

(Refer Slide Time: 20:42)


30. Closure prop of decidable & T.R. languages

decidable	T.R
✓	✓
✓	✓
✓	✓

4. Intersection

L_1 & L_2 are TR via TMs M_1 & M_2 resp.

$L(M) = L_1 \cap L_2$

Input: x

1. Simulate M_1 on x
   If M_1 rej then rej
2. If M_1 acc, we sim M_2 on x
   If M_2 acc then acc else rej

2 Concat

L_1 & L_2 are TR

M_1 & M_2 resp

Input: x

1. Nondet partit $x = x_1 x_2$
2. Simulate M_1 on
   If M_1 rej then
3. If M_1 acc, then
   M_2 on x_2 If M_2
   else rej.

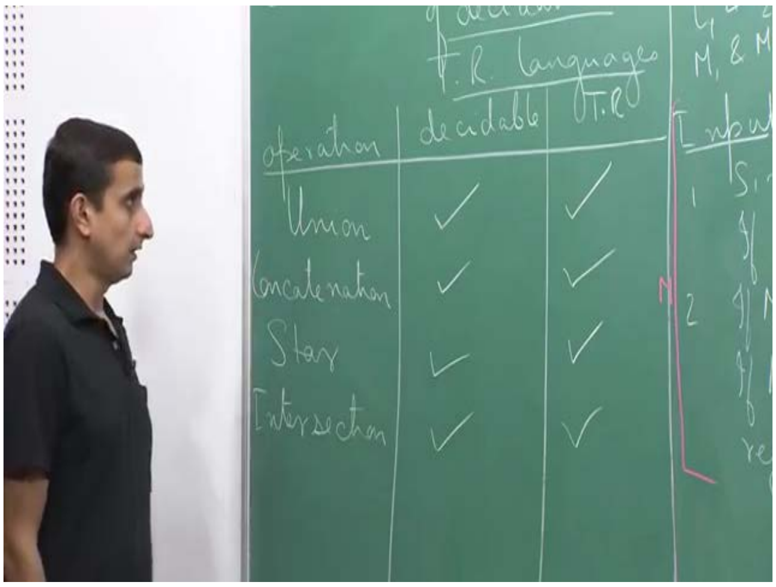
Let us look at the fourth operation that is intersection. So, what happens to intersection? So, remember that of course, regular languages are closed under intersection, but context free languages where not closed under intersection with context free languages. You only know that context free languages are closed under intersection with regular languages. So, what happens in the case of decidable and Turing recognizable languages? Let us look at Turing recognizable.

So, here actually that is the scenario is much better. So, suppose L 1 and L 2 is Turing recognizable via Turing machines M 1 and M 2 respectively. What we do is that we so given an input x we first simulate M 1 an x if M 1 rejects then reject. Next, if M 1 accepts, we simulate M 2 on x; if M 2 accepts then accept, else reject. So, once again observe that the only place where we accept the string x is over here. So, this is the only place where we accept. So, this is our design of a machine M.

And suppose if M 1 goes into an infinite loop. So, if M 1 goes into an infinite loop what it means is that x does not belong to L 1, which means that it should not belong to the intersection of L 1 intersection L 2. Hence, we do not care it just means that M 1 the machine M also does not accept L x, it goes into infinite loop on x.

So, in the cases of intersection, actually we can afford to run the computation in series in sequence, so one after the other. Because here what is necessary is that if x has to be accepted by M, it must first be accepted by M 1, and then it must be accepted by M 2 any other case x is basically either rejected or it goes into an infinite loop. So, this ensures that the language of M is L 1 intersection L 2. Of course for decidable, it is the same argument and it is much easier because both M 1 and M 2 are halting Turing machine.

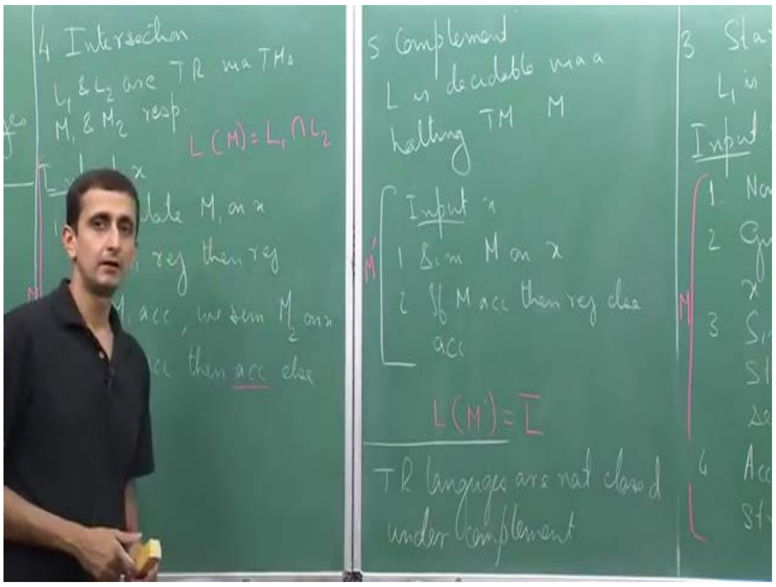
(Refer Slide Time: 24:12)



operation	decidable	T.R
Union	✓	✓
Concatenation	✓	✓
Star	✓	✓
Intersection	✓	✓

So, if we look at the intersection operation both decidable and Turing recognizable languages are closed under them.

(Refer Slide Time: 24:43)

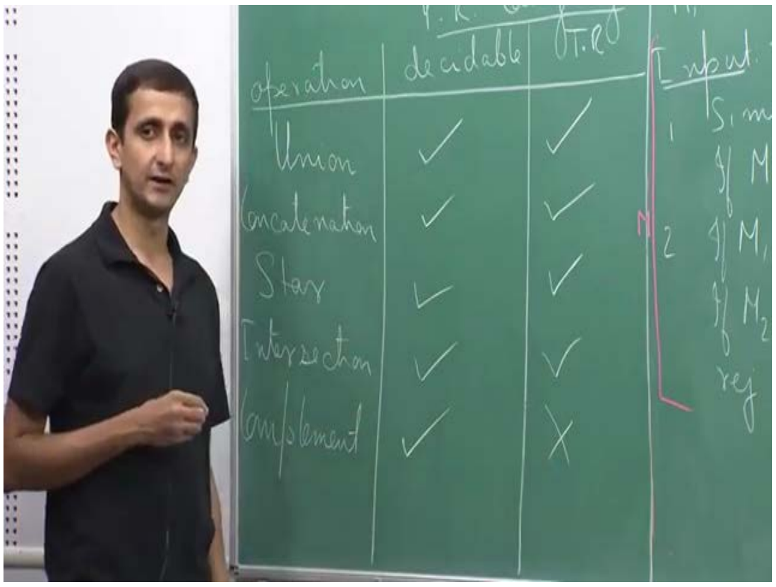


Now we look at the last operation for today that is the compliment operation and here things get a little interesting. So, suppose if L is decidable via a halting Turing machine M, how we can decide L compliment. So, here we are assuming that it is here we are only showing if for the class of decidable languages.

So, suppose if L is decidable then there is a halting Turing machine which accepts L, because we are assuming it is a deterministic Turing machine, there is a unique computation path. So, to accept L compliment, we just have to flip the answer in the end; if M accepts, we reject; if M rejects, we accept. So, we just design a Turing machine which does the following. So, given input x, if so simulate M on x, if M accepts then reject, else accept.

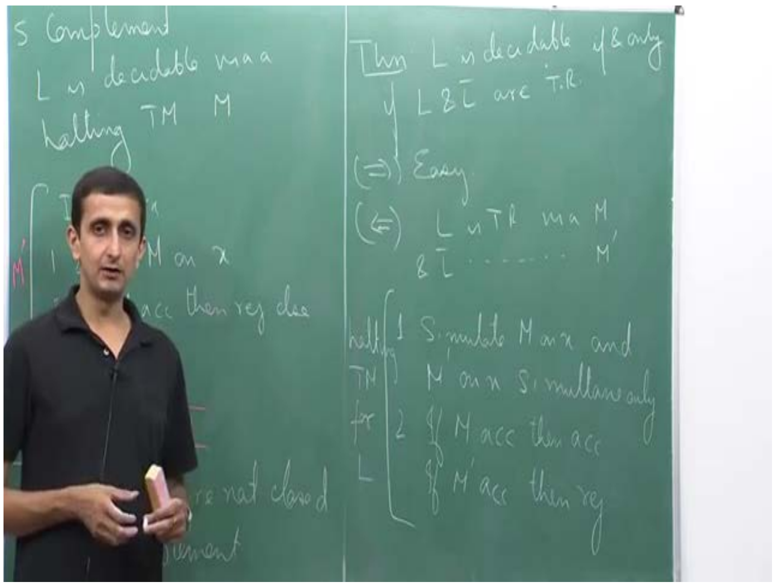
Let us give this machine a name let us say M prime. So, what this imply is that the language of M prime is nothing but L compliment. But what we crucially use in this argument is that the machine M always halts, because if M halts an accept then we are rejecting; if M halts an reject then we are accepting, but that is only through for decidable languages, a halting Turing machines. What if L is a Turing recognizable language?

(Refer Slide Time: 27:01)



So, for complement decidable languages are certainly closed under complement, but as it turns out that Turing recognizable languages are not closed under complement. So, this is an exception. So in fact, we can say a little more, we can say something more about compliment and Turing recognizable languages. So, I will state it as a theorem.

(Refer Slide Time: 28:01)



So, L is decidable if and only if L is Turing recognizable, and L compliment is Turing recognizable case. Let me put it this way. So, L and L complement are Turing recognizable. So, what is the proof? So, suppose one direction is easy; if L is decidable then of course, L is Turing recognizable. And if L is decidable, we just prove that L complement is decidable, which proves that L complement is Turing recognizable. So, hence the forward direction is easy.

Now what about the reverse direction? Suppose, L and L complement at Turing recognizable, how does that imply that L is decidable. So, to do this, actually we construct a Turing machine a halting Turing machine for L which is along the similar lines of the construction of Turing machine for the union case, union of two Turing recognizable languages.

So, what we do is that let say that L is Turing recognizable via M, and L complement is Turing recognizable via M prime. We construct a machine which does the following. So, simulate M on x, if M accepts, so you simulate M on x and M prime on x simultaneously. So, again this is similar to the union as I said. So, you do one step of you simulate x on M for one step, and then you simulate x on M prime for one step, and again you go back to M again you come to M prime and so on. If M accepts then accept if M prime rejects if M is M prime is for L complement. So, if M prime accepts, then reject.

Now, observe that for every string, either M will accept or M prime will accept because every string either belongs to the language or its complement. So, either this guy accept or this guy accept, so you will either be accepting or rejecting. So, on every string, you always halt. So, hence this is a halting Turing machine. So, this is the halting Turing machine for L actually. So, I will stop here.

Thank you.