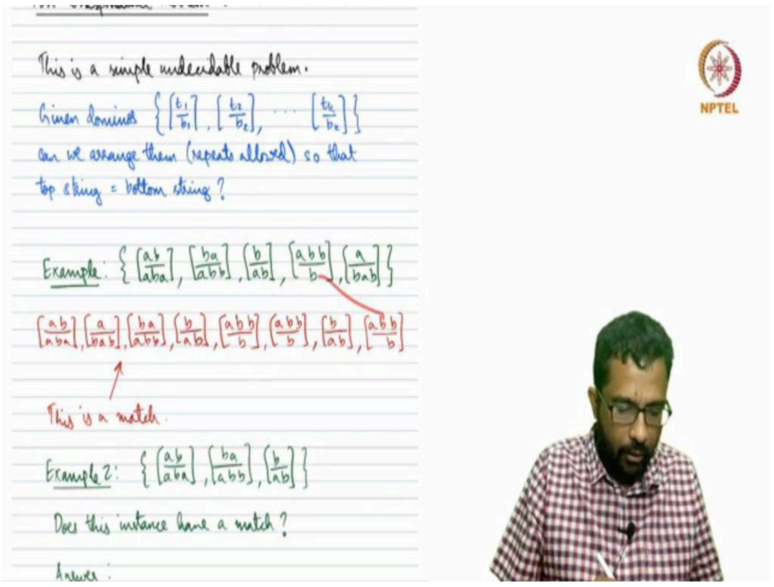
Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
The Post Correspondence Problem

(Refer Slide Time: 00:16)



This is a simple undecidable problem.

Given dominos $\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots \left[\frac{t_k}{b_k}\right] \right\}$
can we arrange them (repeats allowed) so that top string = bottom string?

Example: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right] \right\}$

$\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]$

This is a match.

Example 2: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right] \right\}$

Does this instance have a match?

Answer:

Hello and welcome to lecture 43 of the course theory of computation. This is also the beginning of week 9. So, far we have been seeing decidable and undecidable languages. We saw reductions; we saw how we can prove some languages are undecidable using reductions. And at the end of week 8, I showed you this problem called the post correspondence problem; and I said that this is an undecidable problem.

And the specialty is that so far the undecidable problems that we have seen have been very technical in nature, and have been somewhat abstract. Does this Turing machine halt on the string? Does this Turing machine accept at least a language of this type and so on? So, it is not something that you could describe to a common man or like a school going child. However, the post correspondence problem is a very simple problem.

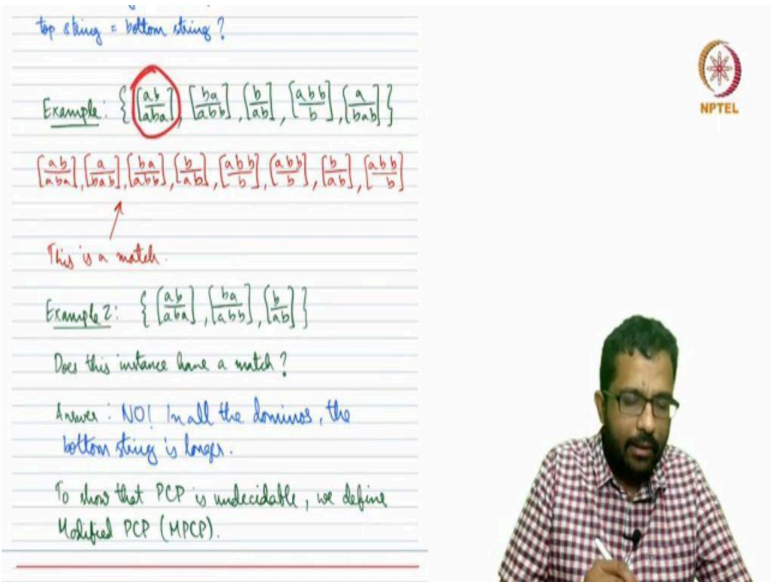
The problem statement can be easily described to anybody. So, the problem statement is the following, given a bunch of dominos, so this is an instance. Is there a way to arrange these dominos such that the top string and the bottom strings match? And the red part this one is a solution for that; this is a match of the instance that is given over here; the instance has only 5 dominos. Now, if you see the solution, the top string ababaabbabbbabb.

So, the bottom string is ababaabbabbbabb , it is the same thing; the top and the bottom are the same. But, of course, the top progresses differently from the bottom; the bottom progresses quickly at first, and then later the top catches on. Some rules I have to mention. First is that, it is not compulsory to use all the dominos; so, not all the dominos need to be used.

And two is that we could use some dominos more than once if we need; there is no restriction on how many times we can use. And then obviously, like we can say the top and bottom strings match if you choose nothing like an empty match. So, the empty match is not usually allowed because that is a that will make every instance have a match.

So, the only two rules are; so we do not need to use all the dominos. And we can use each domino how many of our times that we want; so this has a match that is listed here. And if you see all the dominos is, is a domino that is given in this instance a divided by a bab, ba abb, b and ab over here, abb by b, abb b, then b ab, then ab bb. All the dominos are dominos given here.

(Refer Slide Time: 03:18)



top string = bottom string?

Example: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right] \right\}$

$\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]$

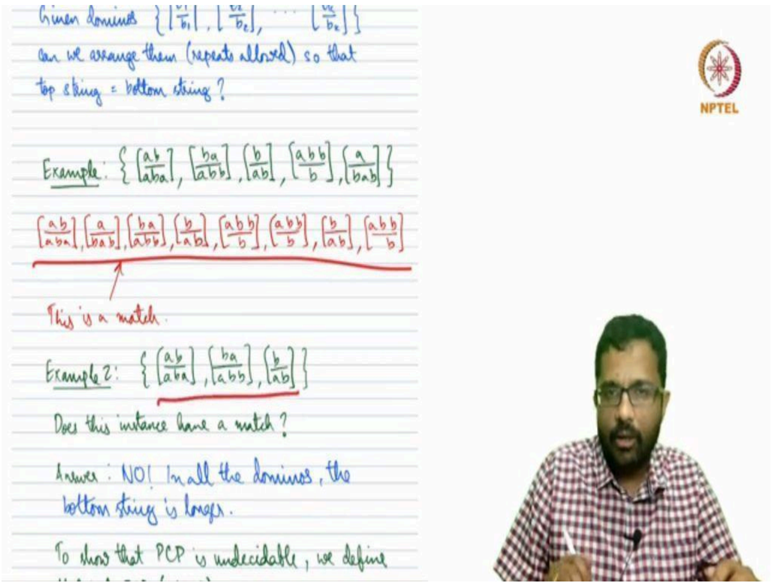
This is a match.

Example 2: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right] \right\}$

Does this instance have a match?

Answer: NO! In all the dominos, the bottom string is longer.

To show that PCP is undecidable, we define Modified PCP (MPCP).



Given dominos $\left\{\left[\frac{a_1}{b_1}\right], \left[\frac{a_2}{b_2}\right], \cdots \left[\frac{a_k}{b_k}\right]\right\}$
can we arrange them (repeats allowed) so that top string = bottom string?

Example: $\left\{\left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right]\right\}$

$\underline{\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]}$

This is a match.

Example 2: $\left\{\underline{\left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right]}\right\}$

Does this instance have a match?

Answer: NO! In all the dominos, the bottom string is longer.

To show that PCP is undecidable, we define

So, another example is does this instance have a match the instance that is given over here? If you think about it a bit, you will see that the answer is no; because in all the dominos, the bottom string is longer. So, how many ever tiles that we arranged? How many ever dominos that you arrange? The bottom string will always be longer and it will continue to keep getting longer. There is no domino that allows the top string to catch on later.

So, if you see the first instance, in fact, the second instance is the same as the first instance; but the last two dominos are not there. It is only the first three dominos of the first incident that are there here. So, here the bottom string is longer; here it gets even longer, here it gets even longer till the fourth domino. And the fifth domino onwards, the first domino starts catching up. But, then there has to be some domino where the first domino is longer than the bottom.

The top domino is longer than the bottom, so that the top can catch up. So here, we have dominos like this where the top domino is longer than the bottom; and the bottom, the top can catch up. However, here in all these dominos, the bottom Domino is the longer one. So, how many of the dominos you put together, the bottom string will always be longer.

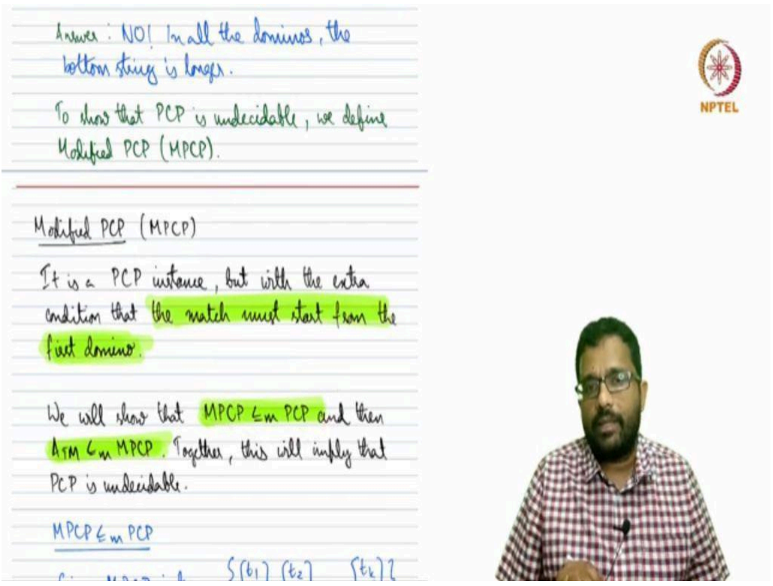
So, this instance does not have a match; so, these are two instances. And the language the PCP question is given an instance like this, or like this; we have to determine whether there is a match or not. In this case, the first instance had a match and the second instance did not have a match; in this case, we could find that. But, is there an algorithmic procedure or is there a standard procedure that can follow and determine whether the given instance has a match?

Turns out, there is no such procedure and this problem is undecidable. So, that is why this problem is interesting; it is a very simple problem. There is no Turing machine; there is whether the EQ $_{CFG}$ or ALL$_{CFG}$, or any such things are not there. It is very simple, it is some dominos; and we have to determine whether there is a match or not. And it is surprising that such a simply describable problem is undecidable.

So, let us move towards the proof of the fact that PCP is undecidable. And in order to prove that PCP is undecidable, we are going to define a slightly modified version of the PCP problem. This will act as an intermediate step. Let us simply call it modified PCP, for lack of our imagination or whatever. Let us call it modified PCP, short MPCP. So, it is like a PCP instance, but with an additional constraint, where we say that it is an instance.

So, the instance is exactly like this, we have to construct a match where the top and the bottom strings match. But, now whenever, so now, I will give you an additional constraint. The conditional constraint is that the match has to necessarily begin with the first domino.

(Refer Slide Time: 06:58)



Answer : NO! In all the dominos, the bottom string is longer.

To show that PCP is undecidable, we define Modified PCP (MPCP).

Modified PCP (MPCP)

It is a PCP instance, but with the extra condition that the match must start from the first domino.

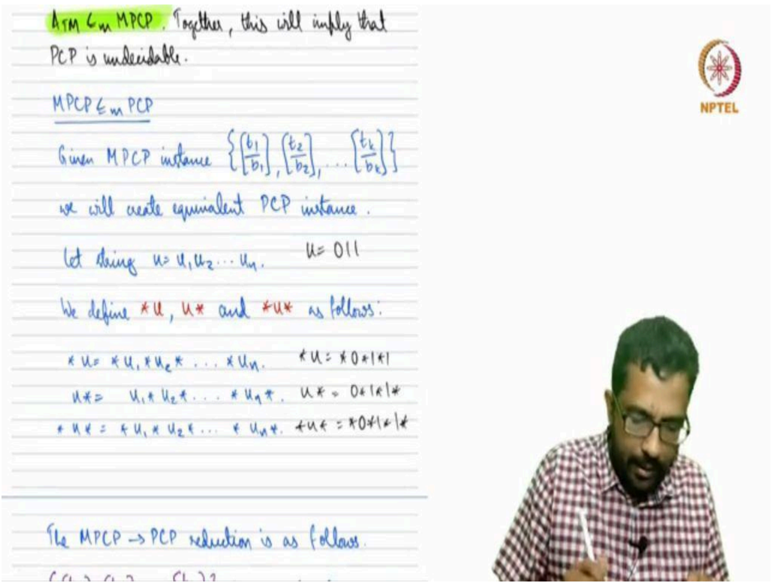
We will show that MPCP $\leq_m$ PCP and then $A_{TM} \leq_m$ MPCP. Together, this will imply that PCP is undecidable.

MPCP $\leq_m$ PCP

So that is the modified PCP, but with the additional condition that the match should always begin with the first domino. So the first domino should be the starting point. So, in PCP, there is no such constraint; I could put whichever domino first. So now, the way we will show that PCP is undecidable is that we will show two things; we will show that first will show that MPCP reduces to PCP, and then A$_{TM}$ reduces to MPCP.

So, A_{TM} reduces to MPCP implies that MPCP is undecidable; and since MPCP is undecidable, MPCP reduces to PCP implies that PCP is also undecidable. So, together these two together imply that PCP is undecided; so this one implies that MPCP is undecidable. And once we have that MPCP is undecidable, this implies that PCP is undecidable. So, let us see the first and let us see the first part MPCP reduces to PCP; this is simpler.

(Refer Slide Time: 08:03)



$A_{TM} \leq_m MPCP$. Together, this will imply that PCP is undecidable.

<u>$MPCP \leq_m PCP$</u>

Given MPCP instance $\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots \left[\frac{t_k}{b_k}\right] \right\}$

we will create equivalent PCP instance.

let string $u = u_1 u_2 \cdots u_n$. $u = 011$

We define $*u$, $u*$ and $*u*$ as follows:

$*u = *u_1 * u_2 * \ldots * u_n$. $*u = *0*1*1$

$u* = u_1 * u_2 * \ldots * u_n *$. $u* = 0*1*1*$

$*u* = *u_1 * u_2 * \ldots * u_n *$. $*u* = *0*1*1*$

The MPCP $\rightarrow$ PCP reduction is as follows.

So, given an MPCP instance, we have to reduce it to PCP instance; so, MPCP the match has to. So, MPCP we are asking if there is a match starting with t1 by b1; so I will call them t1by b1, because so the divide. It is not a division, so it is not like it does not mean divided by; but for want of a better word to describe this, I just call them t1 by b1. So, an MPCP instance means that the match has to begin with t1 by b1. Now, we will create an equivalent PCP instance.

Now, if I replace, if I look at this whole instance as a PCP instance, it need not be equivalent; because in the PCP, there is no constraint that the match starts with t1 by b1. So, it could start with the t_k/b_k for instance, which will not be acceptable in the MPCP instance. So, we will do something in order to force; so we will construct using this instance itself.

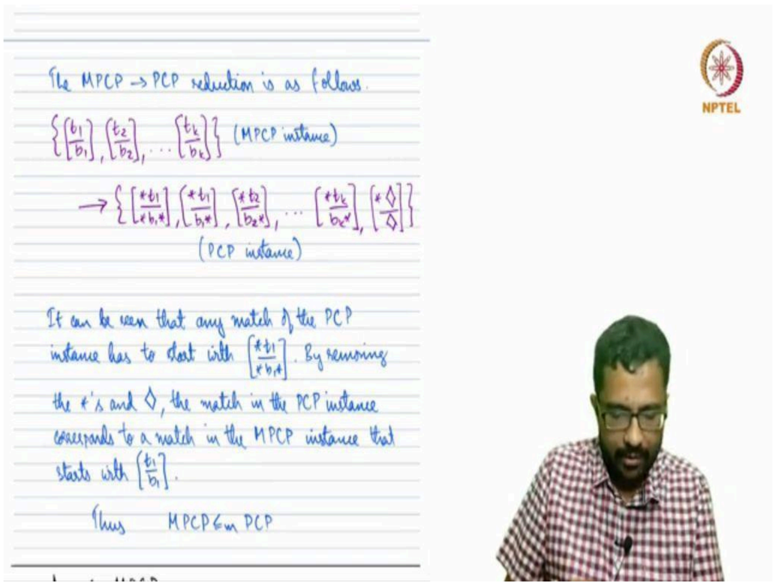
But, we will force to begin with the construction to ensure that a match will begin from t1 by b1, or the corresponding domino from t1 by b1. So, we need to first define some transformations. So notice, so recall that all this t1 t2 up to tk, and b1 b2 up to bk; these are all strings consisting of lots of symbols. So, let u be a string. Now, where u and u1 u2 et-cetera are the individual symbols; let us say that there are n symbols.

Now, I am going to define three notations. One is *u, u*, and * u *; and where *is a symbol that is not there in any of the t1, b1 etcetera, t1t2, b1b2 et-cetera. *is a symbol that is not there in any of these alphabets. So, what is *u? *u means that we start with a star; and before every symbol of u, I am putting a *. So, it is *u1 *, u2 *, u3. So, just to give an example if maybe will use a different color, let say u is 0110.

Then, *u is *0 *1 *1; maybe I will just let us say it is 011. *u is *0 *1 *1; and u *is simply *after each symbol; so it is zero *1 *1 *. There is no *to begin with. In *u, there was no *to end; in u *, there is no *to begin. And finally, *u *has stars at the beginning as well as the end. So, I have this *0, *1 *1 *and stars. So, the last one does not quite look like one; so let me just modify it.

So, this is what I mean by *u, u *, and *u *. So, all of them have stars between each symbol. But *u has one at the beginning, but not at the end. U *has one at the end but not at the beginning; *u *has one at the beginning as well as the end now. Now, using this notation, we will transform the MPCP instance to a PCP instance.

(Refer Slide Time: 11:46)


The MPCP → PCP reduction is as follows.

$\left\{ \left[\frac{b_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \dots \left[\frac{t_k}{b_k}\right] \right\}$ (MPCP instance)

$\rightarrow \left\{ \left[\frac{*t_1}{*b_1*}\right], \left[\frac{*t_1}{b_1*}\right], \left[\frac{*t_2}{b_2*}\right], \dots \left[\frac{*t_k}{b_k*}\right], \left[\frac{*\Diamond}{\Diamond}\right] \right\}$

(PCP instance)

It can be seen that any match of the PCP instance has to start with $\left[\frac{*t_1}{*b_1*}\right]$. By removing the $*$'s and $\Diamond$, the match in the PCP instance corresponds to a match in the MPCP instance that starts with $\left[\frac{t_1}{b_1}\right]$.

Thus MPCP $\leq_m$ PCP

So, this is the MPCP instance, t1 by b1, t2 by b2 et-cetera, tk by bk; the PCP instances are the following. So, notice that the MPCP instance of the first domino has a special position, because the match has to start with this. So, this gets transformed to first *t1 on the top and *b1 *in the bottom; and then we transform all the dominos.

So, the first domino alone gets a special transformation *t1 divided by *b1 *; then we transformed, then every domino gets a transformation. So, let us see what the transformation is. So, t1 by b1 becomes *t1 by b1 *, t2 by b2 becomes *t2 by b2 *. So, there is a *at the beginning, *at the end. tk bk, t *tk bk star; so, this gets this. This transformation is made for everybody t1 b1, t2 b2 and tk bk.

And finally, we add one more new symbol that is a diamond. It is a diamond to get a new domino which is *diamond by diamond. So, the PCP instance has k plus 2 dominos, where the original MPCP instance had k dominos; because the first domino gets a special transformation *t1 by *b1 *. Then, everything else like t1 b1 gets *t1 b1 *, *t2 b2 *, and *tk bk *.

And then finally a special domino to anything *diamond divided by diamond. So, now, let us see what is the specialty here; so, now the point is that. Now, look at this PCP instance, the only domino that we can use to start the match is the first domino here. Because, we cannot start with the second domino; because the second domino top starts with a *, bottom does not.

Same the third domino, same with the tk *, tk by bk *; and same with the *diamond divided by diamond. So, the only domino that we can use to start is this one, and then and then that is okay. So, it starts with *t1 by *b1 *; and then we can append this or this or any other domino, whichever one suits us.

So, any *ti by bi *may follow it. But, notice that with any of these dominos, the issue is that the top ends with the symbol original symbol and the bottom ends with the *. So, it is never a match is never going to end over there. So, because the bottom will end with the *, the top will not. So, the ending thing the only domino that can end the match is this one.

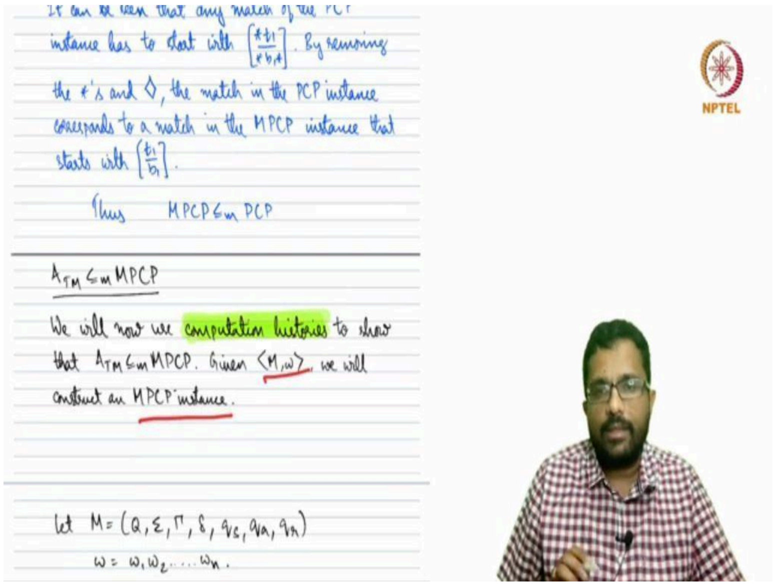
So, we need this diamond to end a match; because that is the only domino where the ending is the same on both top and bottom. So, we end with this at some point whenever the rest top and bottom have matched up, and that constitutes the match. Now, if you remove all the stars and diamonds, what is left is a match of the MPCP with starting with t1 by b1 or t1 by b1.

So, the construction that we have created with the stars and diamonds, basically ensures that we get an MPCP instance but with a PCP match, but starting from t1 by b1. So, that is why this corresponds to a match in the MPCP where we start with the first domino. So, the details I have already told the details, but you can try to work out the details for yourself.

The point is that because of the construction, the PCP instance will have to start with this domino; and when you remove the stars, it corresponds to a match in the MPCP starting with this, which is what is considered a match in the MPCP. So, whenever the PCP instance has a match, each has to necessarily start with this; and hence the MPCP also has a match.

And whenever the MPCP has a match, it is straightforward to see that it corresponds to a PCP match; so that direction is easier. The reverse direction is a bit involved; so, or only slightly more involved and that I have explained. So, this completes the proof that MPCP reduces to PCP and it is very easy to make this modified instance. So, the construction of the modified instance is easy; and the correspondence of the match is also somewhat straightforward. So, now we have to show that A_{TM} reduces to PCP, MPCP sorry.

(Refer Slide Time: 16:46)


It can be seen that any match of the PCP instance has to start with $\left[\frac{*t_1}{*b_1*}\right]$. By removing the $*$'s and $\Diamond$, the match in the PCP instance corresponds to a match in the MPCP instance that starts with $\left[\frac{t_1}{b_1}\right]$.

Thus $MPCP \leq_m PCP$

$A_{TM} \leq_m MPCP$

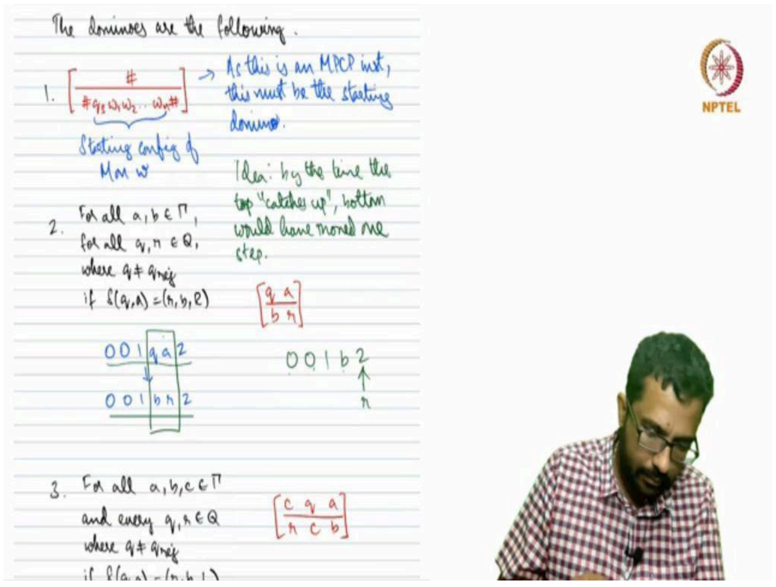
We will now use computation histories to show that $A_{TM} \leq_m MPCP$. Given $\langle M, w\rangle$, we will construct an MPCP instance.

let $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

$w = w_1 w_2 \ldots w_n$.

So, now this reduction uses computation histories. So, now we will show that A_{TM} reduces to MPCP. Given an A_{TM} instance, our goal is to construct an MPCP instance. So, given M comma w, we want to construct an MPCP instance, in such a way that M comma w is an A_{TM} if and only if the MPCP instance has a match. This is an A_{TM} if and only if the MPCP instances a match. In other words, M accepts w if and only if MPCP instances a match.

(Refer Slide Time: 17:25)


The dominoes are the following.

1. $\left[\frac{\#}{\underbrace{\# q_s w_1 w_2 \ldots w_n}_{\text{Starting config of M on w}} \#}\right]$ → As this is an MPCP inst, this must be the starting domino.

Idea: by the time the top "catches up", bottom would have moved one step.

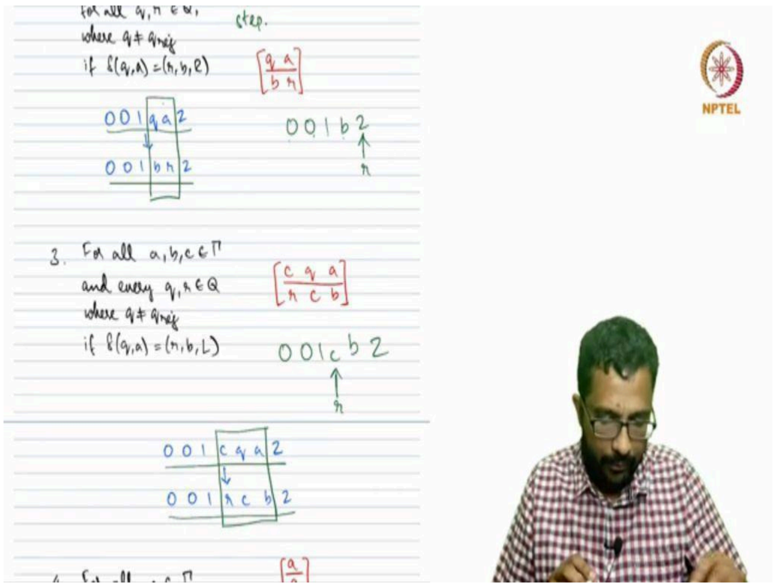
2. For all $a, b \in \Gamma$, for all $q, r \in Q$, where $q \neq q_{rej}$ if $\delta(q, a) = (r, b, R)$ $\left[\frac{q\,a}{b\,r}\right]$

$0\,0\,1\,\boxed{q\,a}\,2$

$0\,0\,1\,\boxed{b\,r}\,2$

$0\,0\,1\,b\,\overset{\uparrow}{2}$ r

3. For all $a, b, c \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{rej}$ if $\delta(q, a) = (r, b, L)$ $\left[\frac{c\,q\,a}{r\,c\,b}\right]$



for all $q, r \in Q$, where $q \neq q_{rej}$ if $\delta(q,a) = (r,b,R)$

step.

$\left[\frac{q\,a}{b\,r}\right]$

$0\,0\,1\,\boxed{q\,a}\,2$

$0\,0\,1\,\boxed{b\,r}\,2$

$0\,0\,1\,b\,2$, $\uparrow r$

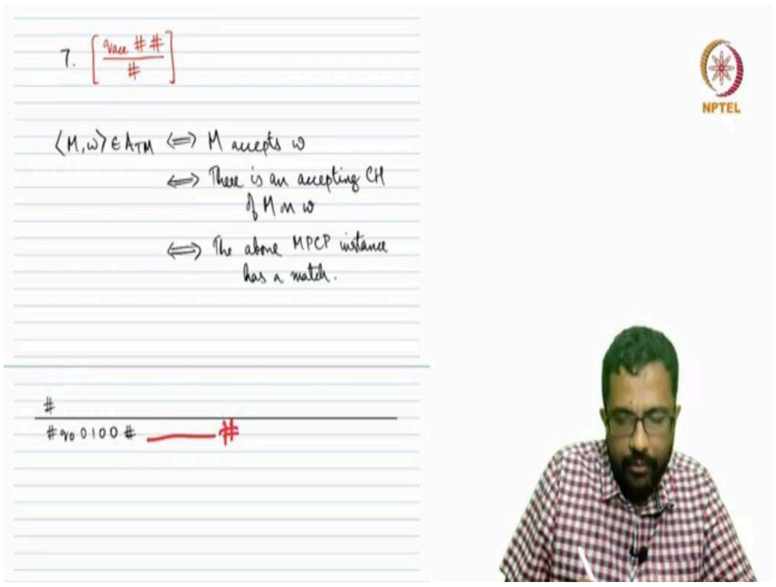
3. For all $a, b, c \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{rej}$ if $\delta(q,a) = (r,b,L)$

$\left[\frac{c\,q\,a}{r\,c\,b}\right]$

$0\,0\,1\,c\,b\,2$, $\uparrow r$

$0\,0\,1\,\boxed{c\,q\,a}\,2$

$0\,0\,1\,\boxed{r\,c\,b}\,2$



7. $\left[\frac{q_{acc}\,\#\#}{\#}\right]$

$\langle M, w\rangle \in A_{TM} \iff M$ accepts w

$\iff$ There is an accepting CH of M on w

$\iff$ The above MPCP instance has a match.

$\#$

$\#\,q_0\,0100\,\#$ —— $\#$

So, let M be defined as Q, sigma, delta, gamma and so on the standard Turing machine. And let the string w be written as w1, w2 up to wn the individual symbols. So, now I am going to describe the MPCP instance; so, the first domino at the top is just a #. And the bottom is #, two #es with this thing in between; and what is this thing?

qs, w1 w2 up to wn. So, in fact, if you observe this closely, you will see that this is the qs to wn is the starting configuration of M on w. So, basically we have the, this is and this is the starting domino in as this is an MPCP instance, this must be the starting domino.

So, now you can see that the starting domino has a # on the top and it starts with a #, has a starting configuration of M on w; and then ends with the # on the bottom. So, the bottom is leading and the top is trailing, and we have a starting configuration. So, it is something like this. So, if I scroll down a bit, it is something like this is what is happening; so, I am referring to this place.

So, when we just have the starting configuration or starting domino, we have something like this. If q0 is a starting state and 011 0100 is the starting input content, then the starting domino gives us this. Now, the goal is to construct dominos such that the, what we will do is I am telling you the high level goal. The goal is to, so now we will have to pick out dominos such that here we get q0 on the top; whatever domino we have to enter will give will should give us q0 here; otherwise, it will not be a match.

But, then the matching thing here may be something else. So, will ensure that by the time the top is filled out completely with the #, this bottom will also get filled out. But, what is filled out here will be the next configuration of the Turing machine; so the idea is the following. So, the idea is so this is starting configuration starting domino; by the time the top catches up, the bottom would have moved one step.

So, when I say one step of the Turing machine, one step of the Turing Machine computation; that is the idea here. So, now when you move one step, and then when you match the top again, the bottom again moves by one step; and finally so the bottom is always ahead. And at the end, we need to somehow make sure that the top is allowed to catch up.

And this happens only when we reach an accept state. And that is why the acceptance of the Turing machine corresponds to the match; so let us see how it happens. Now, the next so the first domino is this that we have explained, # followed by the starting configuration; # on the top, starting configuration in the bottom.

Then, the second set of domino is this, for all a, b in the step alphabet, and q, r in the states, where Q is not the reject state. If there is such a rule that delta q, a gives r, b, R, it is a rule of the Turing machine; then we should add this domino. This domino is the added Domino. So, q, a on the top and b, r in the bottom, so why does it make sense?

So, let us see the tape itself. Let us see how the Turing Machine tape is. Suppose the tape contains this 0 0 1 q a 2, or suppose the configuration is 0 0 1 q a 2; this means that 0 0 1 a 2 and the head points at a, and the state being q. Now, what we are saying is that delta q, a is r, b, R. Meaning we will write b and we will go to state r and move one step to the right.

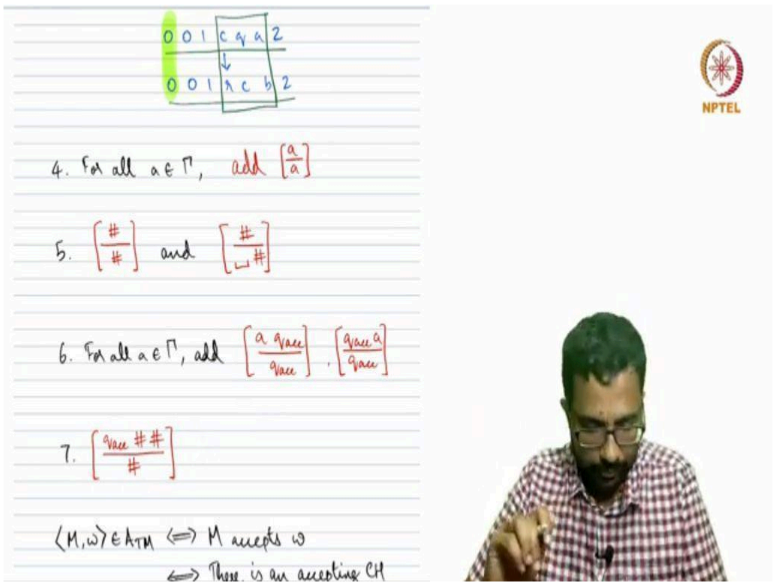
So, this is what delta q, a is equal to r, b, R; means which means now the configuration is 0 0 1 b r 2 which is what I have written over here. So, now everything is the same but I am looking at this part, sorry; I am looking at this part which is the part where it is distinct. q, a in the top and b, r in the bottom; and that I want to capture by the domino which is what I have here.

So, this corresponds to all the moves where the head moves one step to the right. Next thing is for all a, b, c in the tape alphabet and q, r in the state; so same thing, but delta q, a is equal to r, b, L. So, everything is the same, but just that there is a left movement. Then, we have this domino that there c, q, a in the top and r, c, b in the bottom.

Why is that? So, suppose we had 0 0 1 c q a 2. What does it mean? This corresponds to c a 2 with the head pointing at a, and the state being q. So, now delta q, a is equal to r, b, L; meaning you move to, you write b here. You move one step to the left and you go to state r. So, this corresponds to 0 0 1 r c b 2; so, which is what I have here. So, I have this is a configuration that we started with 001 c q a 2; and we ended with 0 0 1 r c b 2.

So, here the difference is in these 3 symbols in the top and the bottom; c q a on the top and r c b on the bottom. So, that is why we have this domino. So, whenever we have rule delta q, a gives r, b, L; we should have this domino like for all the possible c's. And whenever we have this rule delta q, a gives r, b, R we should have this domino. And we should have all like for all such rules, we should have all the possible such dominos. And in this case for all such rules and all possible c's, we should have all possible dominos. Now, the rest of the dominos are fairly simple.

(Refer Slide Time: 26:09)


$$\begin{array}{cccccc} 0 & 0 & 1 & c & q & a & 2 \\ 0 & 0 & 1 & r & c & b & 2 \end{array}$$

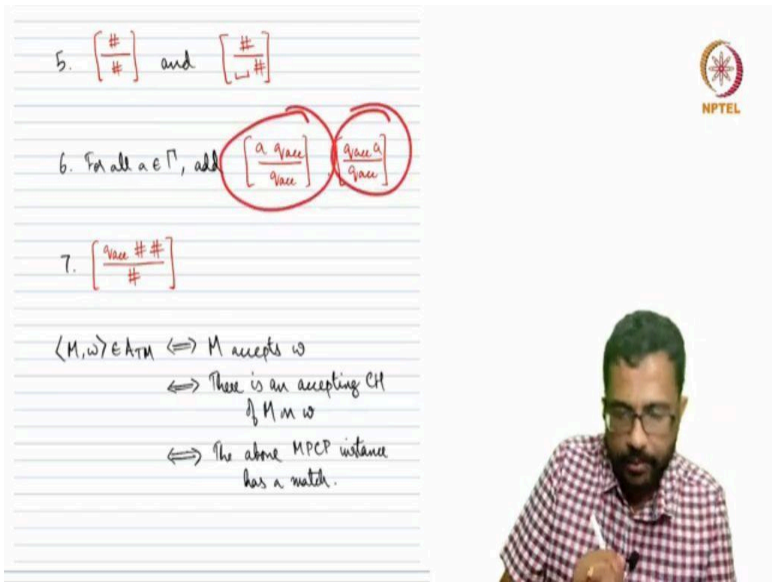
4. For all $a \in \Gamma$, add $\left[\frac{a}{a}\right]$

5. $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{␣\#}\right]$

6. For all $a \in \Gamma$, add $\left[\frac{a\, q_{acc}}{q_{acc}}\right]$, $\left[\frac{q_{acc}\, a}{q_{acc}}\right]$

7. $\left[\frac{q_{acc}\#\#}{\#}\right]$

$\langle M, w\rangle \in A_{TM} \iff M$ accepts w
$\iff$ There is an accepting CH


5. $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{␣\#}\right]$

6. For all $a \in \Gamma$, add $\left[\frac{a\, q_{acc}}{q_{acc}}\right]$ $\left[\frac{q_{acc}\, a}{q_{acc}}\right]$

7. $\left[\frac{q_{acc}\#\#}{\#}\right]$

$\langle M, w\rangle \in A_{TM} \iff M$ accepts w
$\iff$ There is an accepting CH of M on w
$\iff$ The above MPCP instance has a match.

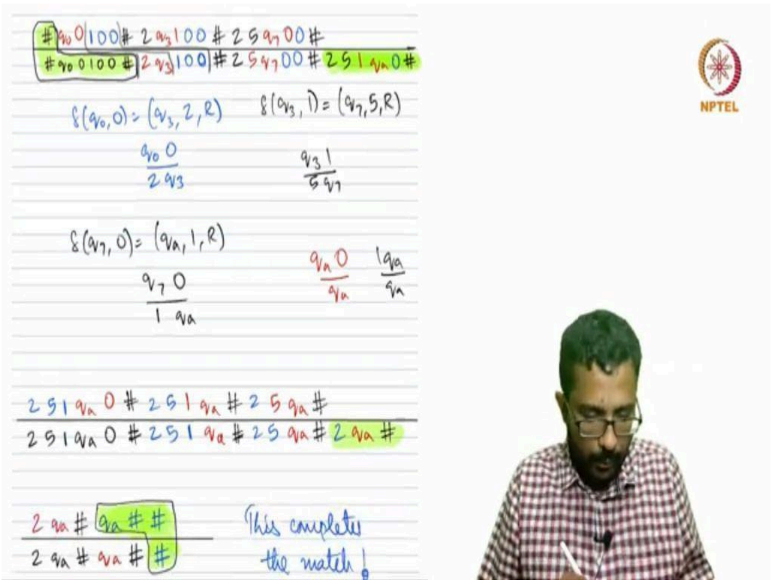
For any symbol a, we add this domino that a is on the top and the bottom. This is for the symbol is the same at the top and the bottom like this one, 001 on the top and 001 on the bottom over here; that is why we have this. And then the next set of dominos has # on the top and bottom; and this is actually for # on the top and blank # on the bottom. This is actually for situations where the head moves to the right.

Then finally, then not finally, for the accepting state, we allow this. You have a q accept on the top and q accept in the bottom; or, you have q accept a in the top and q accept in the bottom. So, a q accept in the top and q accept in the bottom or q accept in the top, and q accept a in the top and q accept in the bottom. So, these are the dominos where we allow the top to catch up with the bottom; and this happens only when we have an accepting state in the top.

So, when basically what is happening is that the accepting state is kind of absorbing the nearby symbols; so, this we write for all possible symbols a. And finally, we have one more domino which is this q accept # # divided by q accept. And this we will see when we work out the example, we will see that this kind of acts as the final step. And the idea of all this is that if M accepts w or if M, w is an A_{TM}, there is an accepting computation history.

And an accepting computation history gives us a match for the MPCP instance. If it is not accepting then there is no match; because we need the accepting state for this absorption to happen. So, perhaps maybe I think the simplest thing is to work out an example and see how these rules work. Right now it may seem like some arbitrary rules defined based on some Turing machine, some set of symbols and transitions; let us see how it works out.

(Refer Slide Time: 28:42)



$$\frac{\#\, q_0 0\, 100\, \#\, 2\, q_3 100\, \#\, 2\, 5\, q_7 00\, \#}{\#\, q_0 0100\, \#\, 2\, q_3 100\, \#\, 2\, 5\, q_7 00\, \#\, 2\, 5\, 1\, q_a 0\, \#}$$

$\delta(q_0, 0) = (q_3, 2, R)$ $\quad$ $\delta(q_3, 1) = (q_7, 5, R)$

$\frac{q_0\, 0}{2\, q_3}$ $\quad$ $\frac{q_3\, 1}{5\, q_7}$

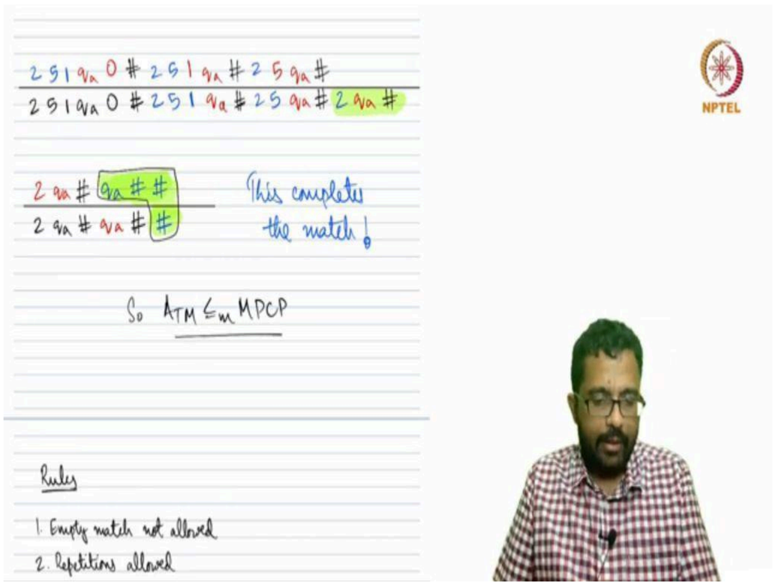
$\delta(q_7, 0) = (q_a, 1, R)$

$\frac{q_7\, 0}{1\, q_a}$ $\quad$ $\frac{q_a\, 0}{q_a}$ $\quad$ $\frac{1\, q_a}{q_a}$

$$\frac{2\, 5\, 1\, q_a\, 0\, \#\, 2\, 5\, 1\, q_a\, \#\, 2\, 5\, q_a\, \#}{2\, 5\, 1\, q_a\, 0\, \#\, 2\, 5\, 1\, q_a\, \#\, 2\, 5\, q_a\, \#\, 2\, q_a\, \#}$$

$$\frac{2\, q_a\, \#\, q_a\, \#\, \#}{2\, q_a\, \#\, q_a\, \#\, \#}$$

This completes the match!



$$\left[\frac{2\,5\,1\,q_a\,0\,\#\,2\,5\,1\,q_a\,\#\,2\,5\,q_a\,\#}{2\,5\,1\,q_a\,0\,\#\,2\,5\,1\,q_a\,\#\,2\,5\,q_a\,\#\,2\,q_a\,\#}\right]$$

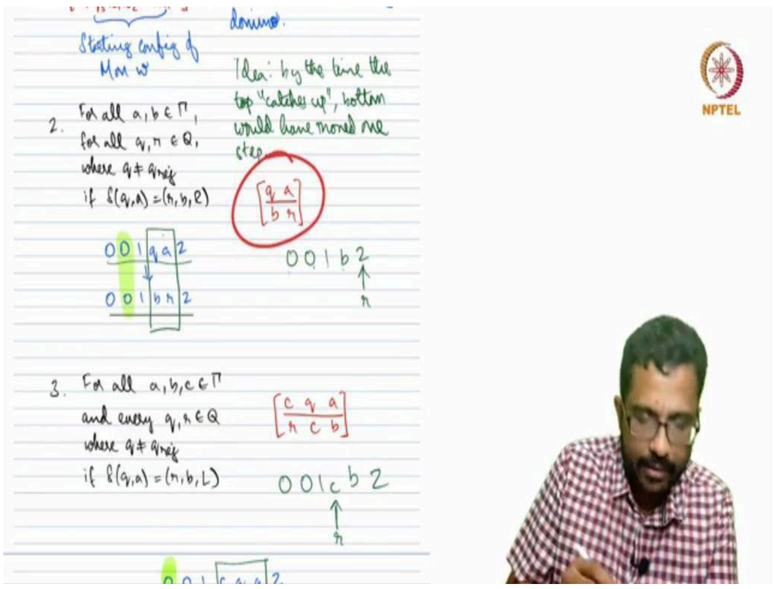
$$\frac{2\,q_a\,\#\,\boxed{q_a\,\#\,\#}}{2\,q_a\,\#\,q_a\,\#\,\boxed{\#}}$$

This completes the match!

So $A_{TM} \leq_m MPCP$

Rules

1. Empty match not allowed
2. Repetitions allowed



Starting config of M on w

domino.

Idea: by the time the top "catches up", bottom would have moved one step.

2. For all $a, b \in \Gamma$, for all $q, r \in Q$, where $q \neq q_{rej}$ if $\delta(q,a) = (r,b,R)$

$$\left[\frac{q\ a}{b\ r}\right]$$

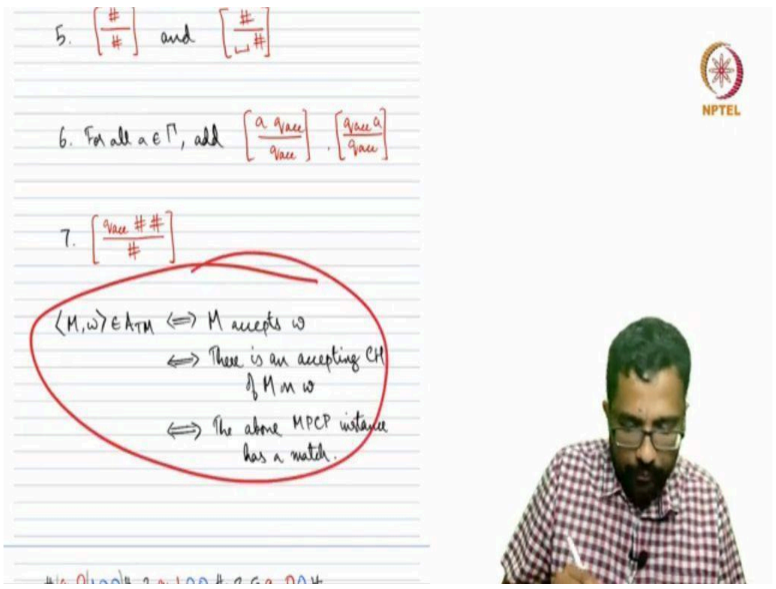
$$\frac{0\,0\,1\,\boxed{q\,a}\,2}{0\,0\,1\,\boxed{b\,r}\,2}$$

$0\,0\,1\,b\,\underset{\overset{\uparrow}{r}}{2}$

3. For all $a, b, c \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{rej}$ if $\delta(q,a) = (r,b,L)$

$$\left[\frac{c\ q\ a}{r\ c\ b}\right]$$

$0\,0\,1\,\underset{\overset{\uparrow}{r}}{c}\,b\,2$



5. $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{\sqcup\#}\right]$

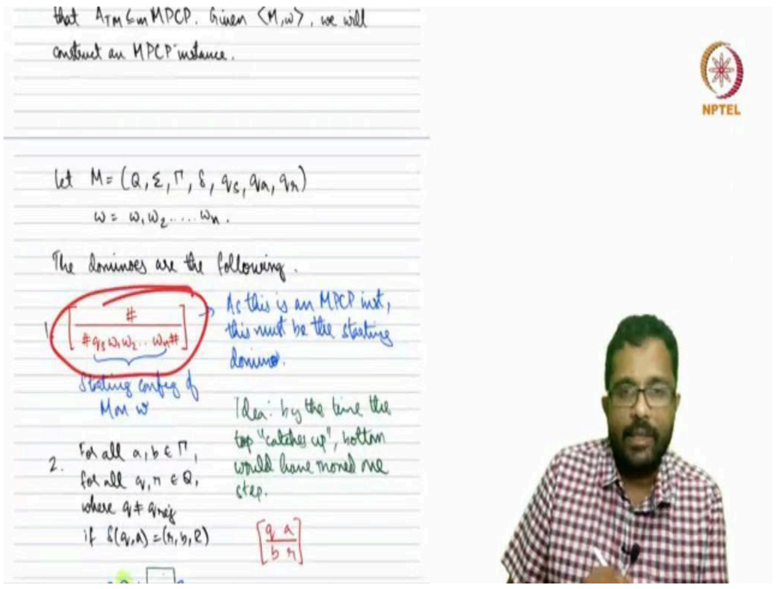
6. For all $a \in \Gamma$, add $\left[\frac{a\ q_{acc}}{q_{acc}}\right]$, $\left[\frac{q_{acc}\ a}{q_{acc}}\right]$

7. $\left[\frac{q_{acc}\ \#\#}{\#}\right]$

$\langle M, w\rangle \in A_{TM} \iff M$ accepts w

$\iff$ There is an accepting CH of M on w

$\iff$ The above MPCP instance has a match.



that $A_{TM} \leq_m MPCP$. Given $\langle M, w \rangle$, we will construct an MPCP instance.

Let $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

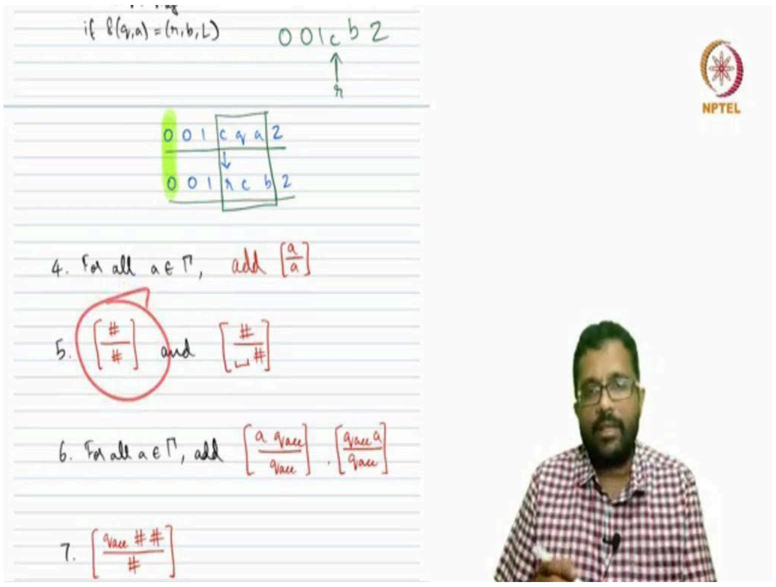
$w = w_1 w_2 \ldots w_n$.

The dominoes are the following.

1. $\left[\dfrac{\#}{\# \underbrace{q_s w_1 w_2 \ldots w_n}_{\text{Starting config of } M \text{ on } w} \#}\right]$ → As this is an MPCP inst, this must be the starting domino.

Idea: by the time the top "catches up", bottom would have moved one step.

2. For all $a, b \in \Gamma$,
for all $q, r \in Q$,
where $q \neq q_{rej}$
if $\delta(q,a) = (r,b,R)$ $\quad \left[\dfrac{q\,a}{b\,r}\right]$



if $\delta(q,a) = (r,b,L)$

$$\begin{array}{c} 0\,0\,1\,c\,b\,2 \\ \uparrow \\ r \end{array}$$

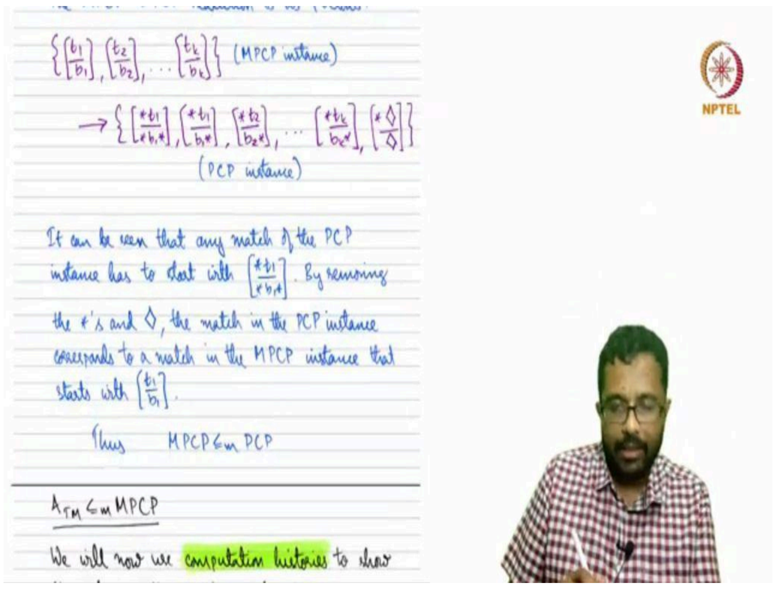
$$\begin{array}{ccc|ccc|c} 0 & 0 & 1 & c & q & a & 2 \\ \hline & & & \downarrow & & & \\ 0 & 0 & 1 & r & c & b & 2 \\ \hline \end{array}$$

4. For all $a \in \Gamma$, add $\left[\dfrac{a}{a}\right]$

5. $\left[\dfrac{\#}{\#}\right]$ and $\left[\dfrac{\#}{\sqcup \#}\right]$

6. For all $a \in \Gamma$, add $\left[\dfrac{a\, q_{acc}}{q_{acc}}\right]$, $\left[\dfrac{q_{acc}\, a}{q_{acc}}\right]$

7. $\left[\dfrac{q_{acc}\#\#}{\#}\right]$



$\left\{ \left[\dfrac{t_1}{b_1}\right], \left[\dfrac{t_2}{b_2}\right], \ldots \left[\dfrac{t_k}{b_k}\right] \right\}$ (MPCP instance)

$\rightarrow \left\{ \left[\dfrac{*t_1}{*b_1*}\right], \left[\dfrac{*t_1}{b_1*}\right], \left[\dfrac{*t_2}{b_2*}\right], \ldots \left[\dfrac{*t_k}{b_k*}\right], \left[\dfrac{*\Diamond}{\Diamond}\right] \right\}$

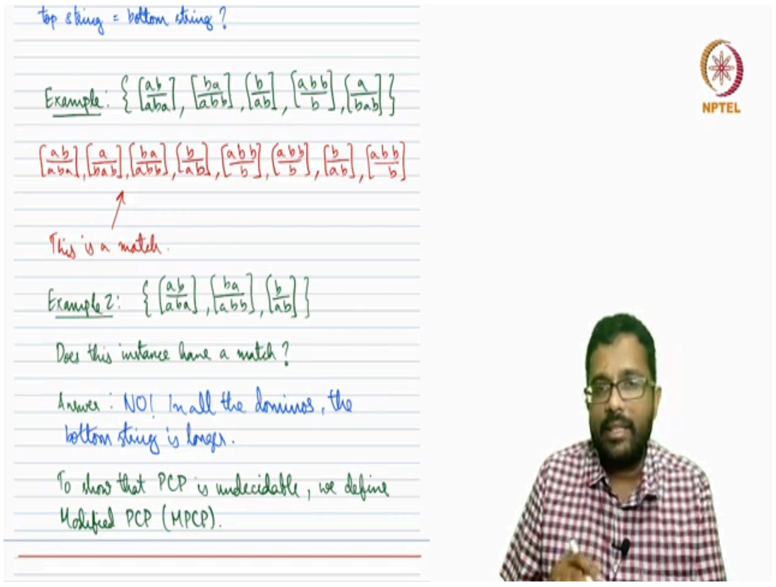
(PCP instance)

It can be seen that any match of the PCP instance has to start with $\left[\dfrac{*t_1}{*b_1*}\right]$. By removing the $*$'s and $\Diamond$, the match in the PCP instance corresponds to a match in the MPCP instance that starts with $\left[\dfrac{t_1}{b_1}\right]$.

Thus $MPCP \leq_m PCP$

$A_{TM} \leq_m MPCP$

We will now use computation histories to show



top string = bottom string?

Example: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right] \right\}$

$\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]$

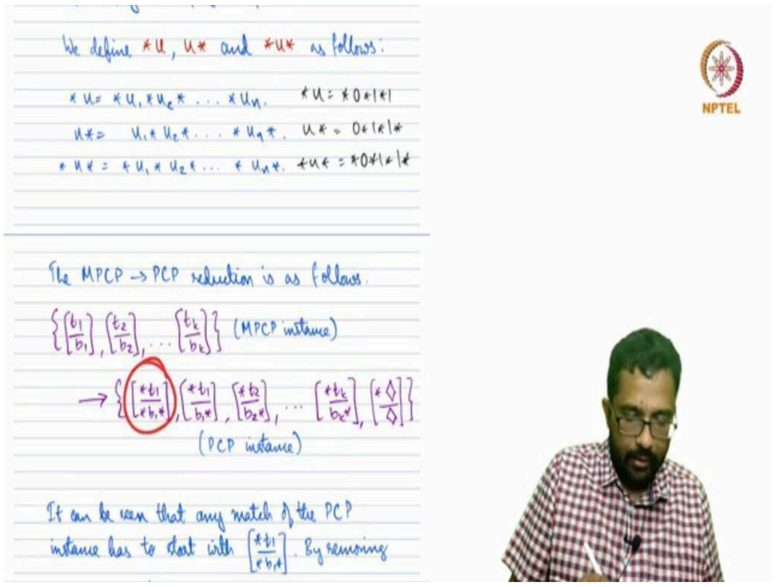
This is a match.

Example 2: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right] \right\}$

Does this instance have a match?

Answer: NO! In all the dominos, the bottom string is longer.

To show that PCP is undecidable, we define Modified PCP (MPCP).



We define $*u$, $u*$ and $*u*$ as follows:

$*u = *u_1 *u_2 * \ldots *u_n$. $\quad *u = *0*1*1$

$u* = u_1 * u_2 * \ldots * u_n *$. $\quad u* = 0*1*1*$

$*u* = *u_1 * u_2 * \ldots * u_n *$. $\quad *u* = *0*1*1*$

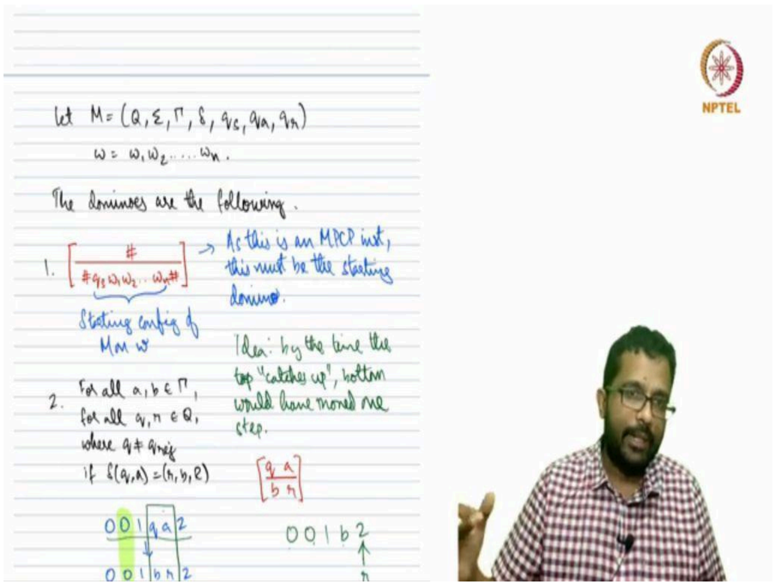
The MPCP → PCP reduction is as follows.

$\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots \left[\frac{t_k}{b_k}\right] \right\}$ (MPCP instance)

$\rightarrow \left\{ \left[\frac{*t_1}{*b_1*}\right], \left[\frac{*t_1}{b_1*}\right], \left[\frac{*t_2}{b_2*}\right], \ldots \left[\frac{*t_k}{b_k*}\right], \left[\frac{*\Diamond}{\Diamond}\right] \right\}$

(PCP instance)

It can be seen that any match of the PCP instance has to start with $\left[\frac{*t_1}{*b_1*}\right]$. By removing



let $M = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$

$\omega = \omega_1 \omega_2 \dots \omega_n$.

The dominoes are the following.

1. $\left[\frac{\#}{\underbrace{\# q_s \omega_1 \omega_2 \dots \omega_n \#}_{\text{Stating config of M on } \omega}}\right]$ → As this is an MPCP inst, this must be the stating domino.

Idea: by the time the top "catches up", bottom would have moved one step.

2. For all $a, b \in \Gamma$,
for all $q, r \in Q$,
where $q \neq q_{rej}$
if $\delta(q, a) = (r, b, R)$ $\left[\frac{q\ a}{b\ r}\right]$

$0\ 0\ 1\ \boxed{q\ a}\ 2$

$0\ 0\ 1\ \boxed{b\ r}\ 2$

$0\ 0\ 1\ b\ 2$

$\uparrow$

r

Now, let us see, so now here this means that q0 is seeing 0; so, let us define some rules. Let us say the rule was delta q0, 0; so it is in states q0, reading zero. Let say q3, let say 2, right. So, q3, 2, right means that we should have this rule acting up. So, we should have the domino, we should have the domino qs, sorry; q0, 0 on the top and 2, q3 on the bottom that domino will be there by the second rule.

So, you can verify that this domino will be, this will give us q0, 0 divided by 2, q3. So, now let me, so since to avoid confusion, I will try to alternate the colours. So, the second domino will have q0, 0 and 2, q3; so, this is in red. Now, the next domino I could just use one on the top and one on the bottom, and 0 on the top and 0 on the bottom, and 0 on the top and 0 on the bottom. And then, let me use # on the top and # on the bottom.

Now, let us say if there is, let us say another rule delta q3, 1 is that q3, 1 is let say q7; I do not know q7, 5, R. So, this corresponds to domino q3, 1 on the top and 5, q7 in the bottom. So, maybe I will do one small thing; I will try to delineate these things. So, the first, sorry; let's say darker colour. So, the first one was this; this was the first domino, this was a; it is too narrow to delineate. So, that is why I have the colour coding here and this is how it is.

Now, I will use something where I have 2 on the top and 2 on the bottom. So, the symbols in the same colour are the ones that match up or that part of the same domino. And then I have q3 on the top; q3, 1 on the top from this domino and 5, q7 on the bottom. And now let me have 0 on the top and 0 on the bottom, 0 on the top and 0 on the bottom; and finally, let me again have # and #. So, now again the situation is that the top is kind of like lagging behind.

Now, suppose it is in state q7 and it is reading a 0. Suppose, delta q7, 0 is it accepts from here; so, qa let say 1 and right. And if that is the case, then we would be q7 0, and the bottom will have 1, q accept. So now, 2 will be 2 here and 5 will be 5. Now, q7, 0 will be on the top and correspondingly here we will have 1, qa; and finally we have 0 here and 0 here.

And finally, we have to match up the #es; so that is the situation. Now, maybe I will just do one thing. We need to move to another line to continue this; push this down so as to make up space. So, now the situation is that I have, let we just carry forward this; carry forward this, this part the highlighted part because that is a place where the bottom is ahead.

So, I will just write that here; sorry, 2 5 1 qa 0 #; and that is where we are. So now, we could use some like we could use 2 here on the top and 2 at the bottom; 5 on the top and 5 on the bottom, 1 on the top and 1 on the bottom. This is which is all like single dominos where you have 2, 2, 5, 5, 1, 1. And then I will use this domino a, q accept by q accept; so, or I will use this one.

So, for instance, I can use this domino q accept 0 by q accept. So, I will just say q accept 0 on the top and the bottom will be q accept. And so let us see what happens now and then let say I use # and #. So, notice that so far, the entire configuration in the bottom was leading; but now the, now again the bottom configuration is leading. But, now the bottom configuration has slightly shrunk; basically, this particular domino allows the top to catch up.

Now, we may use this again; we may use the domino, 1 qa by qa, so let us see. So, first we have 2, 5, 1, maybe I will use another colour; before that, let me just use another colour for this as well. I will use 2, 5, 1 here and 2, 5, 1 here. Sorry, note 2, 5, 1, I will use 2, 5 here and 2, 5 here; sorry, 2, 5, 1 here, 2, 5 here. Then, I will use 1 qa in the top and qa in the bottom; and then again close do the #.

And now I can use 2 on the top and 2 on the bottom, and then I will use 5, qa in the top and qa in the bottom; which is also allowed because anything can go in the top and bottom is just qa. So, now maybe I should carry forward. So, this is a carry forward thing to carry forwarded to qa; let me do it in the next line. I will not need that much space; so I will have 2 qa #.

So, now I can use a single domino where 2 qa is in the top and qa is in the bottom; as I said such a domino is also available as per this rule, two qa is in the top and qa is in the bottom.

And now again I can close it off. So, now this is a situation, the bottom is just leading by # qa. Now, let us see if I can use this one # qa is leading, so we cannot use this right now.

So, first we need to put a # symbol I think. So, we will put a # here and # here; and now the bottom is leading by just qa and #. So, now we can close off things by using this qa # #, and bottom just #. So, I can close off things, maybe I will use a different colour. qa # # in the top and bottom just #; so, basically this is this domino.

So, now you see how we have managed to use; so now that the match is complete as you can see, now the match is complete. So, now it should be clear. So what is happening here? We started off with the starting configuration leading to the bottom and the top was empty. And whenever the top made us caught up in the top, the bottom made one step ahead. So bottom, it corresponds to the configuration of the Turing machine one next step.

So, the top is one step behind in the terms of the configuration of the Turing machine, and this continues; the rules are designed so that this continues till there is a point where the bottom gets an acceptance. And once we have an acceptance, basically then these rule number 6; the dominos in rule number 6 will start to become useful.

Basically, the accepting state can absorb the nearby characters; so that slowly allows the top to catch up. So initially, we had this here 2 5 1 qa 0 in the bottom; so we started off with this extra in the bottom, in the second line. So, now 2 5 1 2 5 1 and the qa 0; the 0 is eaten by the qa. So, the next time we have just 2 5 1 qa; and then the 1 gets eaten by the qa. So, next time we have 2 5 qa; and then 2 gets eaten by the qa.

So, next time we have 2 qa; and finally, sorry, 5 gets eaten; and then we have the 2 also gets eaten. So, finally we have qa and then # and so the bottom is just leading the qa #; and then we just put the domino 7 qa # # divided by #. So, this domino this is the last domino, maybe I will just highlight the last domino, this finishes off things.

This finishes off things. And if you do not get to accept, we will never, we will, the bottom will never be; sorry, the top will never be able to catch up. Because if you can see, the only places where the top is allowed to catch up is when we have an accept state. So, two things should become very clear that whenever there is an accepting computation or the accepting computation history, it results in a match.

And second, if there is no accepting, if there is no accepting computation history or if there is no acceptance, then the top can never catch up. So, this completes this or that is the reasoning for this argument over here. M accepts w if and only if the MPCP instance has a match, and it is also important that this be viewed as an MPCP instance, and not as a PCP instance.

Because, if we view it as a PCP instance, if we do; or in other words, if we do not insist that this be the starting Domino, then there are many many matches. I could just have one tile, # # and # in the bottom; this single tile is a match. Only empty matches are the only thing that are not allowed, this is allowed. So, that is why it is important to have a view of it as a MPCP instance.

And that completes the proof that A_{TM} is reducible to MPCP; so, that shows that MPCP is undecidable. And we have already shown that PCP or MPCP reduces to PCP, so that together implies that PCP is undecidable. So, there you have it, it is a very simple problem; so, given some tiles and dominos. Is there a match?

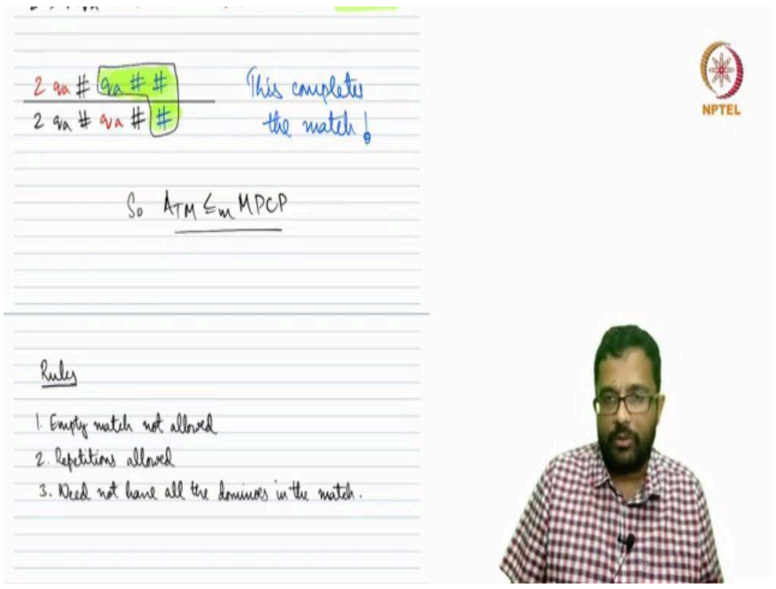
This simple looking question, this innocuous looking question is undecidable. There is no algorithmic procedure by which we can definitively answer this question. That sounds very very surprising, but that is a fact. So, we first reduced MPCP to PCP by putting the *; from the PCP instance, we created a PCP instance. And we ensured that the PCP instance can only start off the match from the first domino; that is how we did the reduction.

Then, for A_{TM} to MPCP, we use computation histories. We created dominos in such a way that the first domino, the top is empty and the bottom has the starting configuration. And then every time we fill out the top to match up with the bottom, the bottom will have advanced one step of the computation. And this goes on; this lead of one step in the computation is maintained throughout till there is an acceptance.

And once there is an acceptance, the tile number 6 allows the top to catch up by absorbing the symbols near the accepting state. And finally we close it with the title number 7; and but then that is available only on acceptance. So, that is maybe I will just also highlight this side; this is the starting configuration here starting domino here, and we end with this term.

And that completes the match; and it is now fairly clear that the MPCP instance is a match if and only if M accepts w; or in other words if and only if M, w is an A_{TM}. That completes the reduction and hence we can see that PCP is undecidable. That is all I have for this lecture.

(Refer Slide Time: 45:19)



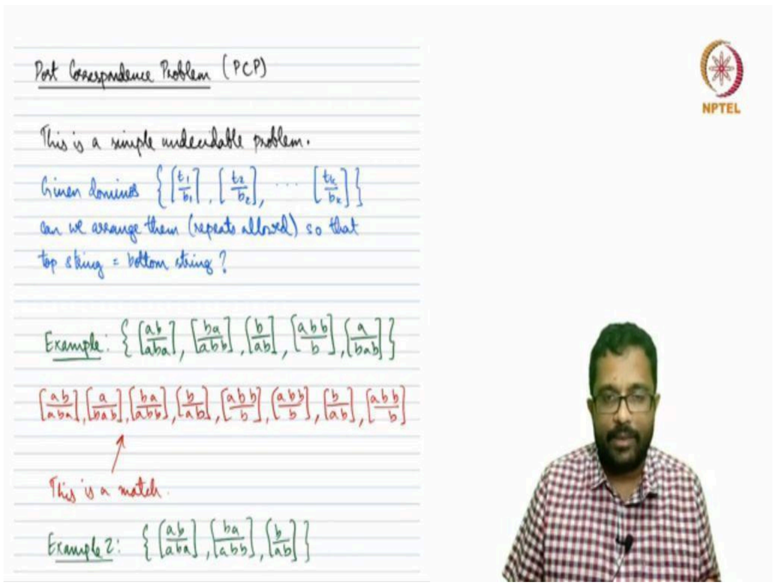
$$\frac{2\, q_a \# \boxed{q_a \# \#}}{2\, q_a \# q_a \# \boxed{\#}}$$

This completes the match!

So $A_{TM} \leq_m MPCP$

Rules

1. Empty match not allowed
2. Repetitions allowed
3. Need not have all the dominoes in the match.



Post Correspondence Problem (PCP)

This is a simple undecidable problem.

Given dominos $\left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots \left[\frac{t_k}{b_k}\right] \right\}$

can we arrange them (repeats allowed) so that top string = bottom string?

Example: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{a}{bab}\right] \right\}$

$\left[\frac{ab}{aba}\right], \left[\frac{a}{bab}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right], \left[\frac{abb}{b}\right], \left[\frac{b}{ab}\right], \left[\frac{abb}{b}\right]$

This is a match.

Example 2: $\left\{ \left[\frac{ab}{aba}\right], \left[\frac{ba}{abb}\right], \left[\frac{b}{ab}\right] \right\}$

And these are some rules that I have just written down anyway. One is that an empty match is not allowed; otherwise, it is meaningless. Two is that repetitions are allowed. And three is that we need not have all the dominos in a match, necessarily like we could skip some dominos or many dominos also. And all of these I have said earlier, but I just noted it down. And that completes the proof that PCP is undecided; so it is a problem that is undecidable.

And it is a very very simple problem; so that is the surprise and that is the contrast as well. So far, all the undecidable, other undecidable problems were somewhat technical and abstract; this is very very simple and easy to describe. And that completes the proof that PCP is undesirable; and that is also the end of lecture 43. In lecture 44 we will just see a brief application of the fact that PCPs are undecidable. So, see you in lecture 44. Thank you.