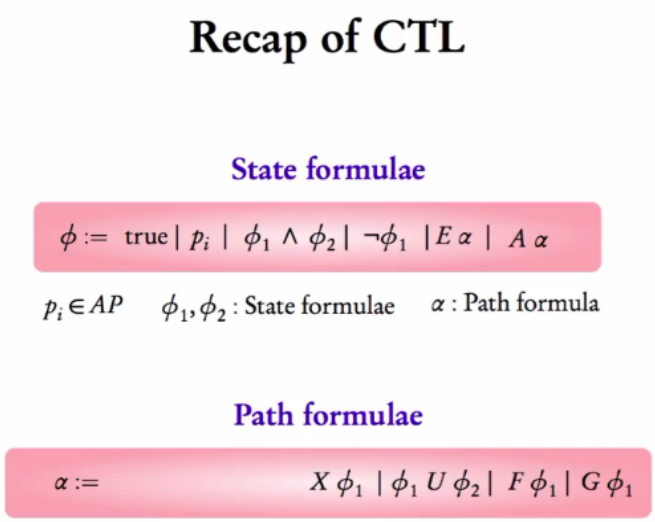
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 45
Adequate CTL formulae

Welcome to unit 10 of this course. In the last unit, we saw computation tree logic. This was a logic used for reasoning about properties of the computation tree of a transition system. In this unit, we will be looking at algorithms for CTL. In the first module, we will try to reduce the generic CTL formula into some kind of a normal form. We will try to re-write every CTL formula into some special form.

(Refer Slide Time: 00:48)



Let us first recap the syntax of CTL. CTL is made of two kinds of formulae, if you remember, state formulae and path formulae. So state formulae are evaluated over the states of a transition system. True, the state formula true is true in every state. pi, this could p1, p2, p3, p1 is true on states, which are labelled with p1. So every state is labelled with a set of atomic propositions and if that set of atomic propositions contains p1, then the state formula p1 is true on that state.

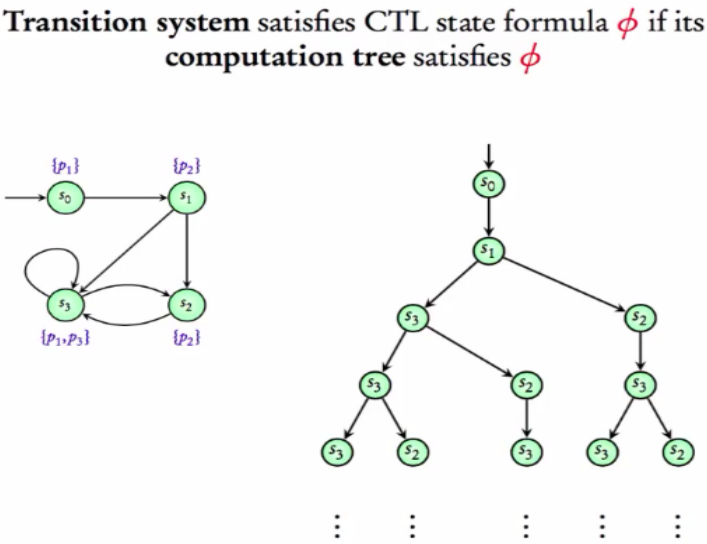
Now, what about phi 1 and phi 2, so phi 1 and phi 2 are state formulae. If both of them are true on a state, then this formula is true. Similarly, if phi 1 is not true on a state, then negation of phi 1, not phi 1, this formula is true in that state. These are the new things introduced in

CTL. This says that there exists a path from this state, which satisfies a certain property given by the path formula alpha.

This state formula says that from this state, if you look at all possible paths all of them satisfy the path formula alpha, and to reason about paths we have used these temporal operators. This says that in the path, the next state satisfies phi 1. This formula says that in the path, so this is evaluated on a path, this is evaluated on a state. So phi 1 until phi 2 being true on a path means that phi 1 is true till a point where you see phi 2.

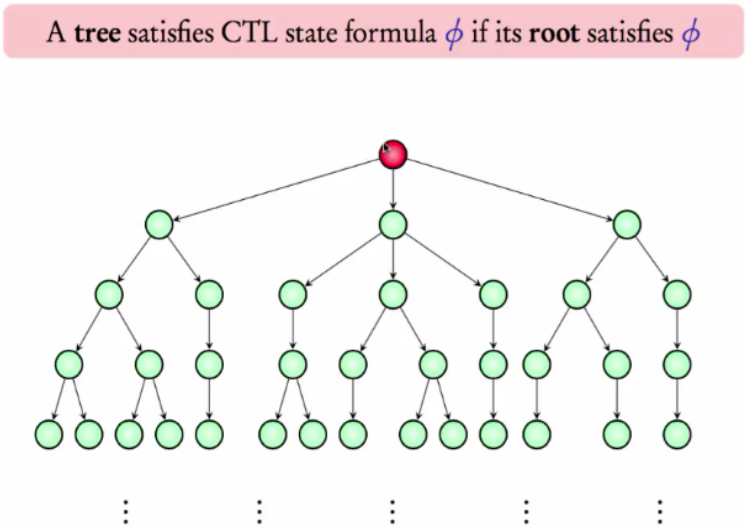
And, of course, you know what X phi 1 is at some point of time, phi 1 is true on the path and G phi 1 says that everywhere, every state in this path satisfies phi 1, okay. This was a quick recap of CTL.

(Refer Slide Time: 03:30)



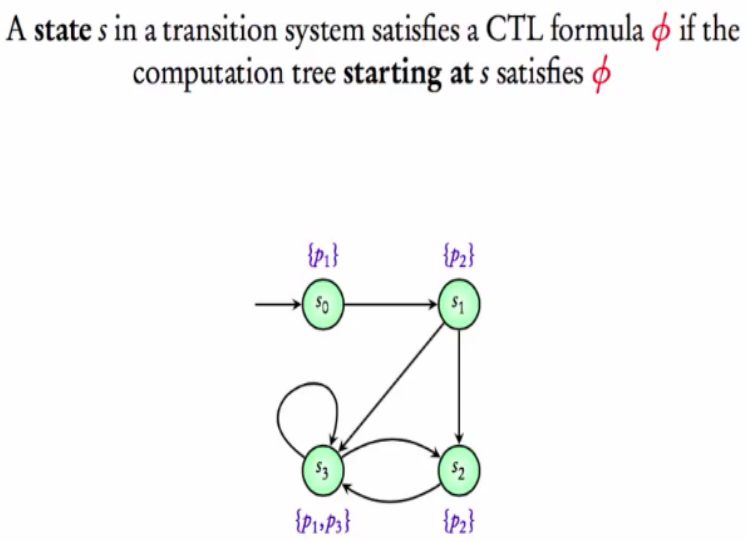
When does a transition system satisfy a CTL state formula phi, so we say that a transition system satisfies a CTL state formula phi, if its computation tree starting from the initial state satisfies the CTL formula phi. Now when do you say that this tree satisfies phi.

(Refer Slide Time: 04:02)



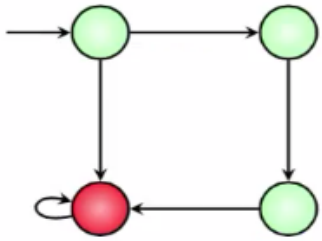
The tree satisfies CTL formula phi, if its root satisfies phi, okay. So essentially a transition system satisfies a CTL state formula phi, if its root or rather if its initial state satisfies phi, okay.

(Refer Slide Time: 04:22)



You can also evaluate a CTL state formula on some state of the transition system, not necessarily the initial state. We can evaluate the state formula in this state, say, in that case we look at the computation tree starting from that state, and the root of that computation tree should satisfy the state formula phi, okay.

(Refer Slide Time: 04:52)

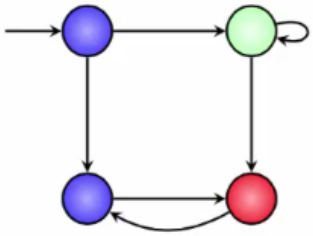


Above transition system satisfies E X *red*

Let us look at examples, look at this transition system, this state is coloured red. Look at the initial state of the transition system, if you look at the computation tree starting from this initial state, there exists a path, okay, look at this path which keeps looping such that, the next state in that path satisfies red. If you imagine the computation tree of this transition system, this would be the root and these two would be the children at the first level.

So there exists a path which satisfies X red, correct. So this transition system satisfies EX red.

(Refer Slide Time: 05:56)

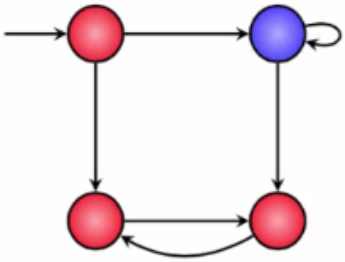


Above transition system satisfies E *blue* U *red*

Let us look at this transition system, look at this path, okay, there exists a path where blue until red is true. So try to imagine the transition, sorry, the computation tree of this transition system, this will be the root, these two will be the children of the first level, and then this will be a child of this, and this will again be a child of this, this will also be a child.

So if you look at the tree starting from the root, you will see that there exists a path where a blue until red is true. So we say that the initial state of the transition system satisfies E blue until red. Hence, the transition system in itself satisfies E blue until red.

(Refer Slide Time: 07:04)



Above transition system satisfies E G *red*

Look at this, there exists a path where all the states are red, correct, so E G red is true in this state. Hence, the transition system satisfies E G red. Note that it does not satisfy A F blue, let us see what it means, what is A F blue, A says that all paths satisfy the property F blue. So starting from this initial state, if you look at all paths, do they satisfy the fact that every path reaches a blue state, no, because this particular path never reaches a blue state. Hence this transition system does not satisfy A F blue.

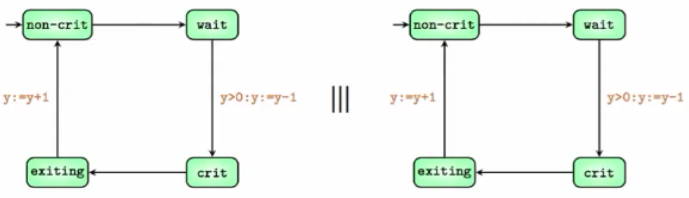
(Refer Slide Time: 08:19)

Mutual exclusion

Atomic propositions AP = { p_1,p_2,p_3,p_4 }

p_1: pr1.location=crit p_2: pr1.location=wait

p_3: pr2.location=crit p_4: pr2.location=wait



Above system satisfies **A G** ¬ (p_1 ∧ p_3)

Recall the mutual exclusion example, which we had seen last time as well. We checked that this transition system satisfy the property that in every path, every state satisfies not of p1 and P3, that means it is never the case that both process one and process two are in its critical section, okay.

(Refer Slide Time: 08:52)

Goal of this unit

Design an **algorithm**:

INPUT: A **transition system** M and a **CTL formula** ϕ
OUTPUT: Does M satisfy ϕ?

We will answer a more general question:

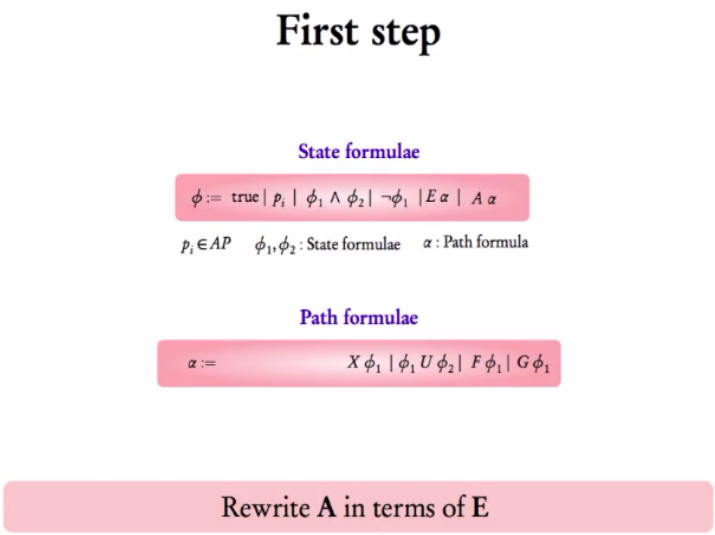
Given M and ϕ, find all the states of M that satisfy ϕ

So that was a short recap of CTL and examples of CTL properties on transition systems. So what is the goal of this unit, CTL is a logic, which reasons about infinite paths. However, given a finite transition system and a CTL formula, how do you get an algorithm which says whether the transition system satisfies the formula phi. That is going to be the subject of this unit.

We will design an algorithm, which takes as input a transition system M and a CTL formula phi, and its output would be, yes if M satisfies phi, and it will be no if M does not satisfy phi. We will actually answer a more general question, so given M and a CTL formula phi we will find all the states of the transition system that satisfy the CTL formula phi. And then this question reduces to asking, whether the initial state is in this set of states, okay.

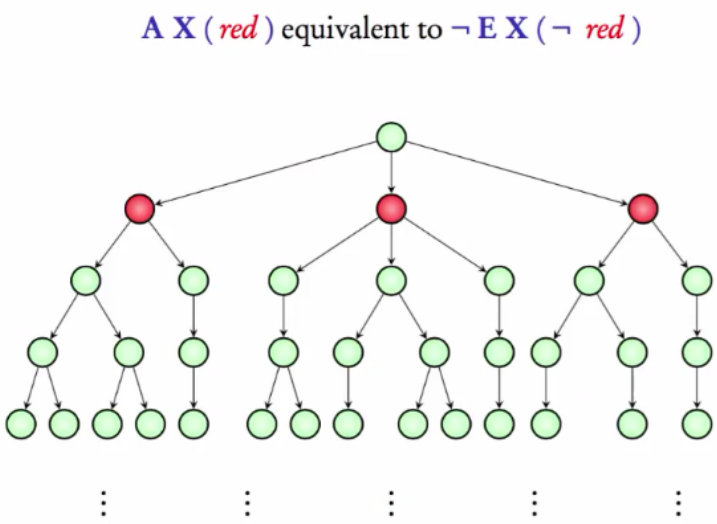
So for instance, in this transition system this state also satisfies E G red, similarly this state also satisfies E G red. So we will ask given this and E G red, what are the set of states that satisfies this formula, then the algorithm should tell me this, this, this not this, because this is marked blue, so any path out of it will not satisfy G red, because this state itself is not red, okay. So my algorithm should tell me that this state, this state and this state satisfy E G red.

(Refer Slide Time: 11:15)



So as a first step towards giving this algorithm, we will try to simplify the CTL formulae, by simplification, I mean that we will try to rewrite every CTL formula in terms of just E, actually we want to rewrite formulas using A in terms of E.

(Refer Slide Time: 11:46)

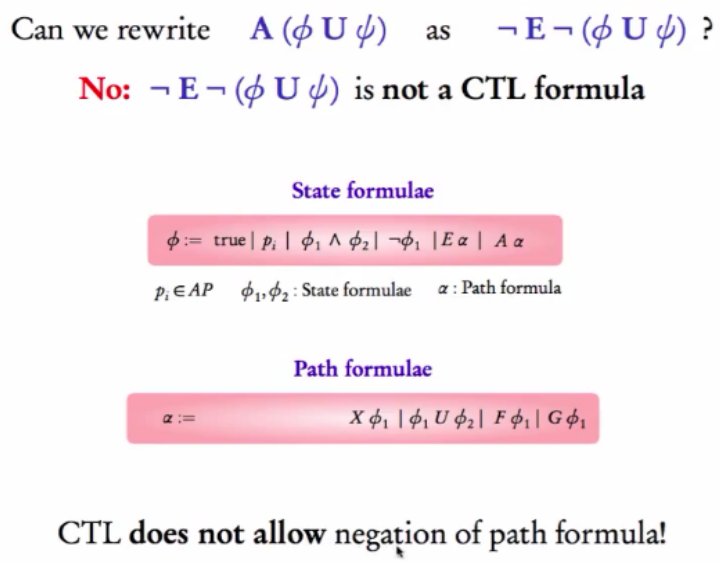


Let us see, A X red, look at this computation tree, A X red means, you look at all paths from the initial state, in this case the root, you look at all paths from the root every child of the root should satisfy red. That is what it means, A X red in every path the next state that is the second state should be red. This is equivalent to same that there does not exist a path, where the next state is not red, correct.

Let me repeat it again, there does not exist a path in which the next state is not red, correct. So A X red can be rewritten using just E. All paths satisfying a certain property is the same as

saying that it is not true that there exists a path where the does not satisfy that property, okay. This is what is written here A X phi, whenever you see the formula A X phi you can rewrite it as not of E X of not phi, try to get this clear before we get into the next step.

(Refer Slide Time: 13:40)



What about A of phi until psi, here we were talking about A X phi, suppose we have A of phi until psi, which says that every path satisfies phi until psi. So can we rewrite it as, it is not the case that there exists a path which does not satisfy this formula. Well, the meaning is the same, this formula and this formula mean the same, however this thing not of E of not phi until psi, was not a CTL formula.

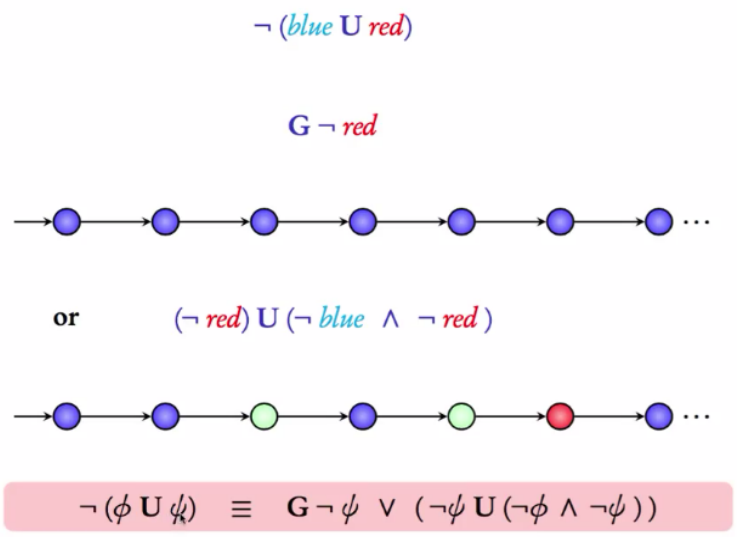
This is a CTL star formula, let us see why, how does a CTL formula look like. If I have an E, let us try to derive this, if E of alpha, alpha can be of the form phi 1 until phi 2. Here my alpha has not of something. This is not allowed by CTL. All the nots should come before state formulae. So E of not of phi until psi cannot be derived using this syntax. Hence, we should find a different way of rewriting A of phi until psi.

This is what it says. CTL does not allow negation of a path formula. If you remember, this is a legal CTL star formula, but not a CTL formula.

(Refer Slide Time: 15:41)

Let us now try to write A of phi until psi, so when I write A U, it just means that A of phi until psi, we will rewrite it using E U and E G. I mean, we will rewrite it in terms of some formula, which uses exist E of phi 1 until phi 2 and E G phi 3, okay. Let us see how we are going to do it.

(Refer Slide Time: 16:09)



Let us first try to understand the LTL formula not of phi until psi. Here I have taken phi to be blue and psi to be red. Let is slowly understand what it means to say that blue until red is not true. One possibility, the entire path has no red. Then blue until red cannot be true, right. So blue until red will be true only if the path has a red somewhere and blue has to be true till that point.

So if that is not true, then it means that either the entire path has no red, which can be written as G of not red or the other option is the path has a red, however, blue is not true till that point. In fact, for it to be true, blue has to be true here as well. See, here we are using colours, but in general they can be atomic prepositions and each state can have multiple atomic prepositions as well, okay. I hope you notice that.

If blue until red is not true, either the entire path has no red in it or it has a red, however, before the point where it becomes red, blue is not true always, okay, which means there exists a point before this place where it becomes red, which satisfies not of blue, let us now describe this path using an LTL formula. What does this say? The path has a red here, however, there exists a point before the place where it becomes red and that point does not satisfy blue, correct.

So what does this translate to? Not red is true until a point where not blue and not red are true, okay. This is the first point where red was there. Before that itself blue should be away and in particular this place does not satisfy blue and does not satisfy red as well and every point before that does not satisfy red. See not red is true all over the place till here and what we want to claim is that in the path where not red is true, blue becomes false as well.

That is what is said by this formula. So not of blue until red is equivalent to either G of not red or not of red until not blue and not red. Hence, we can rewrite not of phi until psi as G of not psi or not psi until not phi and not psi. Do not get scared by this notation. It just explains the intuitive understanding that we developed here. Put psi to be red and phi to be blue, you will get this formula, okay. Please go through this till you understand it because it is an important step.

(Refer Slide Time: 20:50)

$$\mathbf{A}\,(\phi\,\mathbf{U}\,\psi)$$

$$\equiv$$

$$\neg\,\mathbf{E}\,\neg\,(\phi\,\mathbf{U}\,\psi)$$

(Not a CTL formula)

$$\equiv$$

$$\neg\,(\,\mathbf{E}\,\mathbf{G}\,\neg\,\psi\;\vee\;\mathbf{E}\,(\neg\psi\,\mathbf{U}\,(\neg\psi\,\wedge\,\neg\,\phi))\,)$$

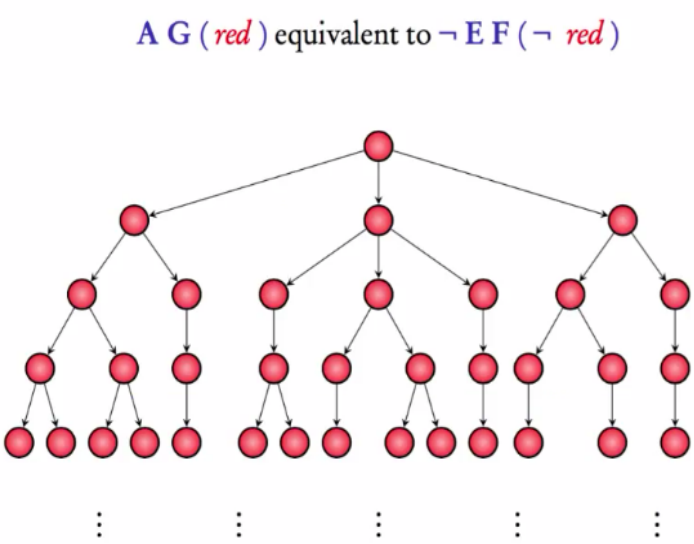
(A CTL formula!)

Now, our goal was to rewrite A of phi until psi in terms of E. This is what we started off with. We wanted to write it as, it is not the case that there exists a path, which satisfies not of phi until psi. However, this was not a CTL formula. So now, knowing that not of phi until psi can be rewritten in a different form, which can use that. So what we need to say is that, see this not is here. We want to say that there exists a path that satisfies not of phi until psi, either there exists a path of this form or there exists a path of this one. That is, it.

So E of not of phi until psi is written like this. Either there exists a path where everywhere not psi is true or there exists a path where not psi is true until not psi and not phi become true. This is a legal CTL formula because here the nots are with the state formulae. E of some phi 1 until phi 2 and here this is a state formula, this is a state formula, so this is phi 1 or phi 2 and this entire thing is a state formula in front of it is a not.

As an exercise, figure out how you can derive this using the CTL syntax. I just explained it, but you try to do it yourself. So we have written A of phi until psi as a CTL formula. So far we have managed to rewrite Ax and Au in terms of E.

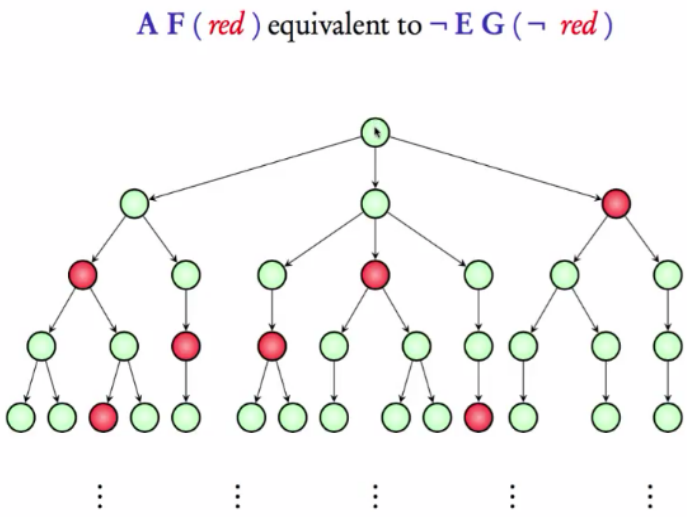
(Refer Slide Time: 23:05)



Let us now look at A G, AG red. This is simple. AG red says that every path, in every path, every state, satisfies red. It is equivalent to saying that it is not the case that there exists a path where not red is reached sometime, okay. So if you forget this not, this says that EF of not red, there exists a path where not red is reached. In that case, AG red will not be true, so that is why you put the not.

So all paths satisfy red in every state translates to saying that it is not the case that there exists a path where not red becomes true.

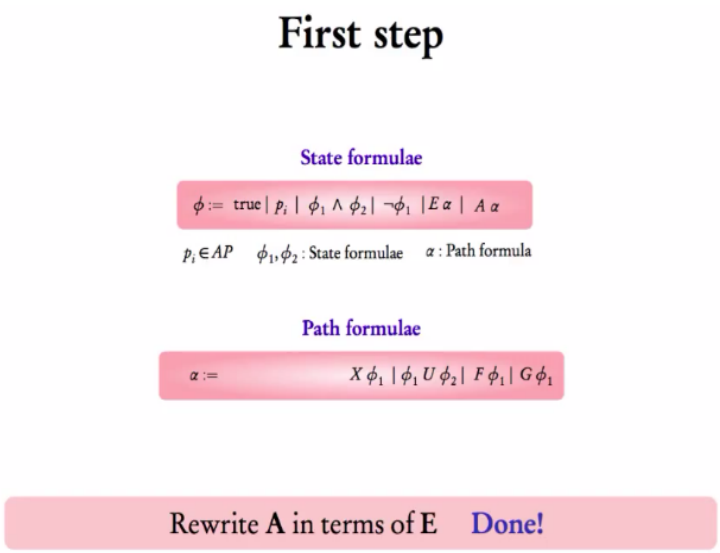
(Refer Slide Time: 24:04)



Finally, AF red, this states that in all paths, you can reach a node, which is coloured red. This is equivalent to saying that it is not the case that there exists a path where every state is coloured not red, yeah. This is the same, the dual of the F would be G. All paths reach

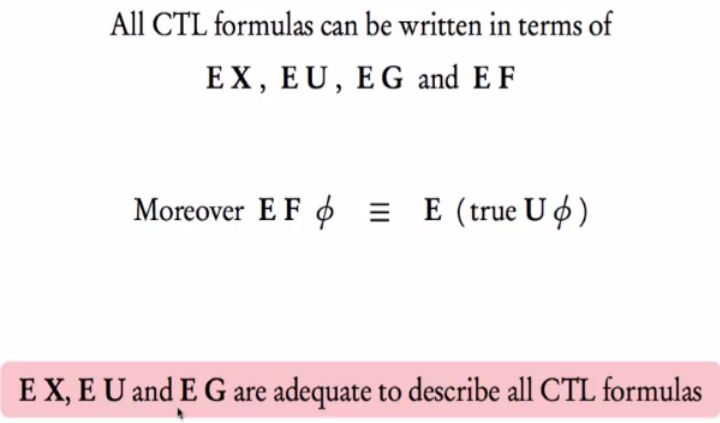
something is the same as saying it is not the case that there exists a path where everything satisfies not of the property, okay.

(Refer Slide Time: 24:47)



So we have managed to rewrite A in terms of E. We rewrote Ax A of phi 1 until phi 2, AF phi 1 and AG phi 1 in terms of just formulas using E, correct. Yeah, this is done.

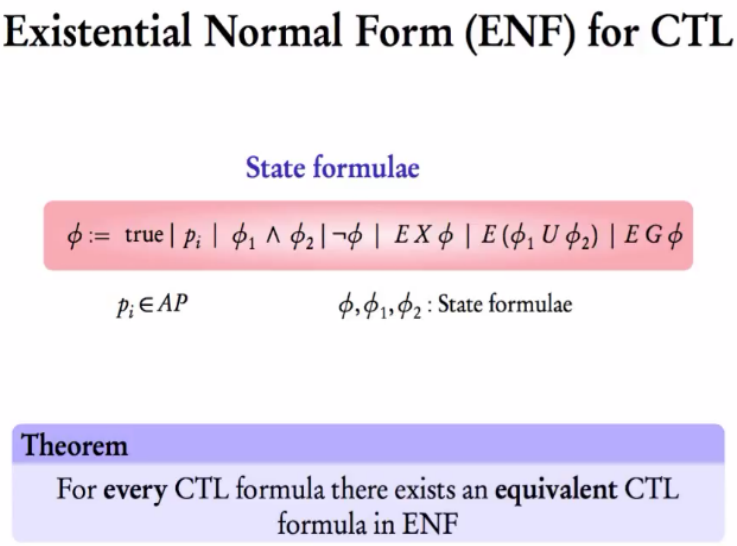
(Refer Slide Time: 25:14)



All CTL formulas can be rewritten in terms of EX, EU, EG, and EF. Because whenever, see a CTL formula is made up of this syntax. Whenever you have A of say phi 1 until phi 2, you rewrite it using E as we did in a previous line. So wherever there is AX phi 1, replace it with not of E of X of not phi 1, okay. Similarly, for the other things. Let us do a bit more of pruning. Look at EF, EF phi is just E of true until phi, because of F is just true until phi, right.

So we can even remove EF. We can say that EX, EU, and EG are adequate to describe all CTL formulas. See, the whole syntax of CTL is useful when you want to write properties. You want to say every path satisfies something and so on, so for that it is useful to keep A, however, for an algorithmic purpose, we can rewrite A in terms of E, so that it is enough to talk about algorithms for EX, EU, and EG.

(Refer Slide Time: 26:49)



So this is called existential normal form for CTL. So this is the entire CTL for you, either true phi 1 and phi 2. Phi 1 and phi 2 are state formulas, you can do this. If phi is a state formula, you can take not of that, then you can say there exists a path, which satisfies X of phi. You can see there exists a path, which satisfies phi 1 until phi 2 and then you can say there exists a path satisfying G of phi, okay.

Note that you can do nesting, you can say EX, EG P1. You can say E of EG P1 until T2, say, okay. So these kinds of things you can do, just that this normal form will be useful for us in giving algorithms. So this is the theorem for this module. For every CTL formula, there exists an equivalent CTL formula in E and F. In the next modules, what we will do is, we will give algorithms for EX, EU, EG and from this we can derive the formula for the entire CTL.