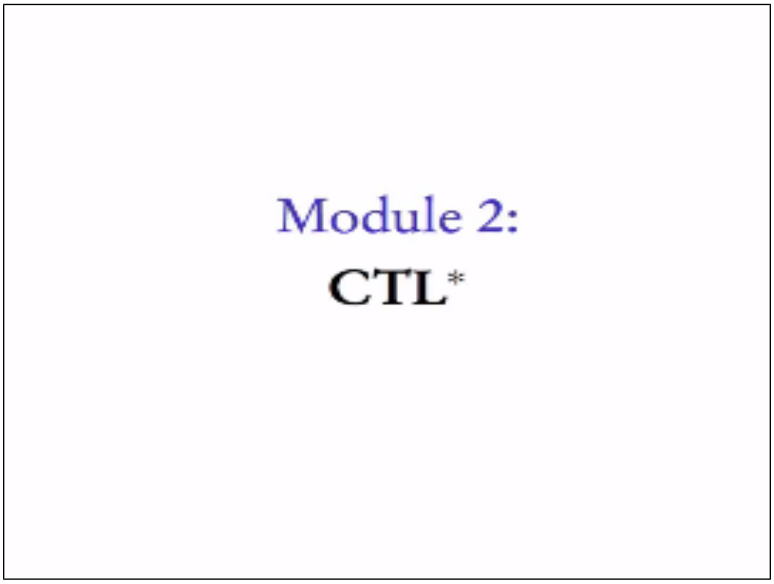
**Model Checking**
**Prof. B. Srivathsan**
**Department of Computer Science and Engineering**
**Indian Institute of Technology – Madras**

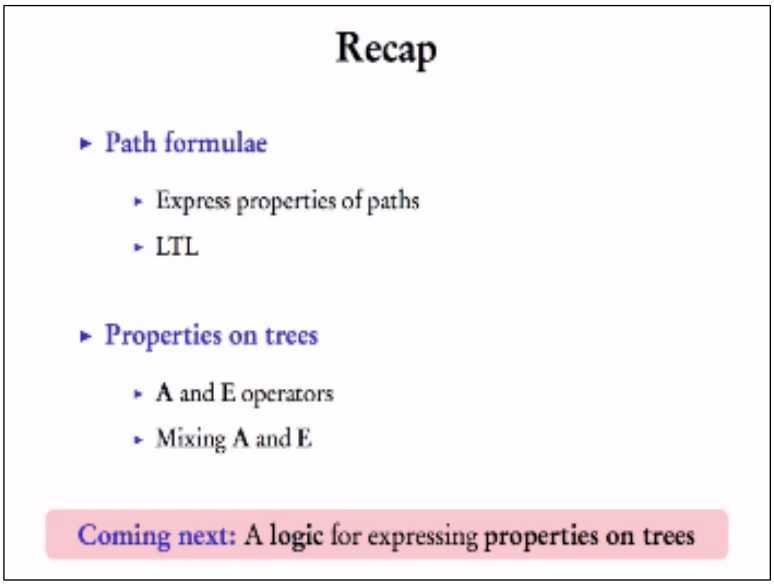
**Lecture - 42**
**CTL***

**(Refer Slide Time: 00:01)**




In this module, we will be looking at a logic for expressing property is over computation trees. It is called Computation Tree Logic star, CLT*. Let us start off with CTL *.

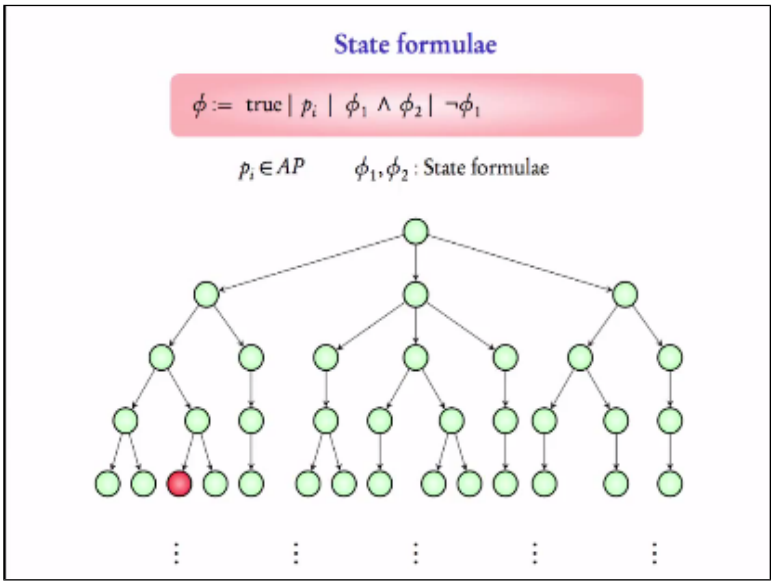
**(Refer Slide Time: 00:25)**



A short recap of what we have seen so far. LTL is a Logic that can express properties of paths. So a formulae of LTL can be called path formulae. LTL is just one way of expressing properties of paths; we could have generic path formulae to express more general properties of paths. To carry them over onto trees we introduced the A and E operators for all paths and exists a path and we also saw a way of mixing A and E to get interesting properties on trees.

We will now define a logic for expressing properties on trees.

**(Refer Slide Time: 01:24)**

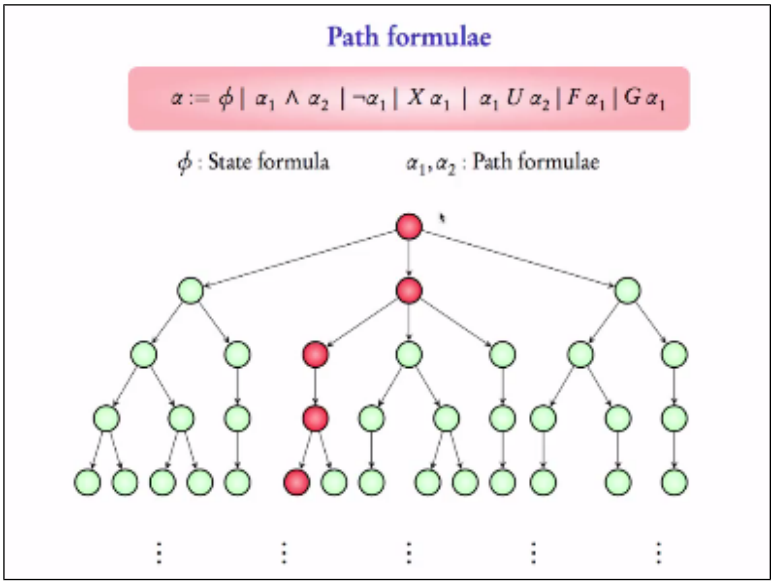



Look at this tree. Each node in this tree is some state of a transition system and it is labeled with atomic propositions. Consider some state, what are the things that you can say on a state. So these are called state formulae. Let us see. You can evaluate true on a state; every state satisfies true. For every atomic proposition each proposition Pi is a state formula. You can evaluate it on a state, so this state satisfies red.

Similarly look at this state each state-- you look at the label of that state suppose of the label of the state is P1, P2 then this state satisfies the state formula P1. These are characteristic of states. Given two state formulae suppose I know that some state formula is true on this and some other state formula is true on that state you can make an AND of this two phi 1 and phi 2, similarly you can make a NOT as well.

So these are the things that you can evaluate on state. Two is true on every state. P1 will be true on the state which have the label P1 similarly P2 would be true on the states which are label by P2 you can form ANDs and NOTs you can say P1 AND NOT P3 for instance. Okay. These are all properties which can be evaluated on the state itself. And hence they are called state formulae.

**(Refer Slide Time: 03:29)**




Let us now look at path formulae, the things that you can talk over paths. So Alpha is a path formula if it is of the following forms. Suppose you look at a state formula you can also evaluated on a path and say that this path satisfies this formula phi if the initial state satisfies this state formula. For example, look at this state formula P1 AND NOT P3 we can say that this path satisfies P1 NOT P3 if this initial state satisfies P1 AND NOT P3.

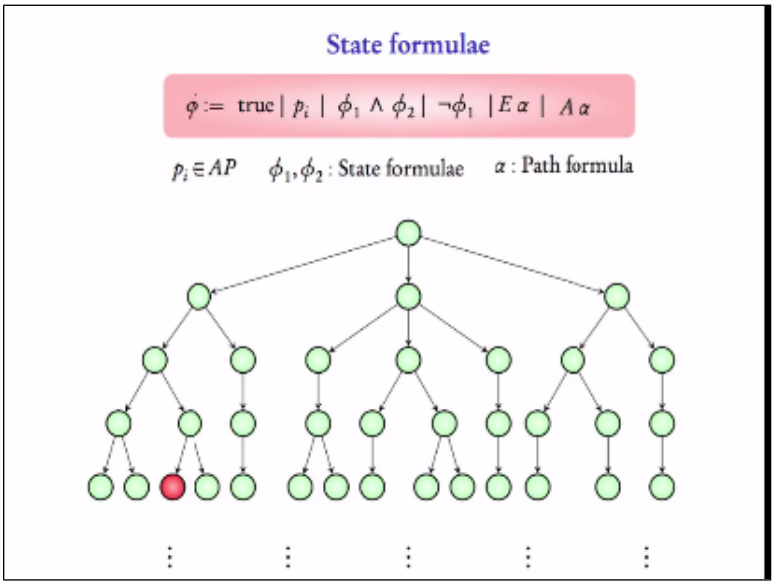
This is the same way we defied LTL formulae. So every state formula can be looked at as a Path formula as well and the meaning of that is that the first state satisfies this formula. If you have two Path formulae you can make AND and NOT of it. If this path satisfies both Alpha one and Alpha two then you can you say that it satisfies Alpha one and Alpha two. Suppose this path does not satisfy Alpha one then you can say that that it does not-- I mean it satisfies NOT of Alpha one.

These are the temporal operator. Now start the temporal operators. You can say on this path that X of Alpha one is true that means the path starting from position satisfies Alpha one this is same like LTL. You can also say Alpha one until Alpha two. So these are all path; these are all

evaluated on paths Alpha one until alpha two X of Alpha one and so on. Look at also talk about F and G which are derived out of until.

But still let me write then down as X Alpha one G Alpha one just for clarity. So you can look at a path this path satisfy G (red) similarly some other path will satisfy F of some (P1) and so on. So these are all formulae that can be evaluated on paths.

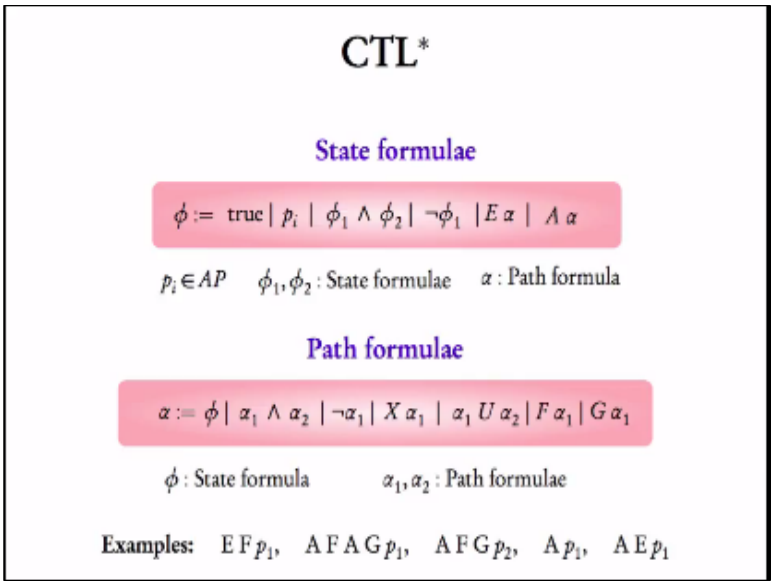
**(Refer Slide Time: 06:07)**




Now let us get back to State formulae. So these were the state formulae P1, P1 AND NOT P2, P1 AND P3 and so on. We saw path properties were do A and E sit. You can say E of Alpha on a state, so given a state you can say that E of a path formula is true on this state. That means there exists a path that satisfies Alpha. So E alpha is actually a property of the state. It is the characteristic of this state. So E Alpha is a State formula. You need to understand this.

These are all easily State formulae. What about E alpha? Look at this state, you can evaluate E alpha on the state and E alpha will be true on the state if there exists a path that satisfies Alpha. Okay. Similarly, E Alpha is again a state formula because it is the property-- it is the characteristic of the state. All paths starting from the state satisfy Alpha. So path formulae are the ones which involved the temporal operators X until F and G, and state formulae have this quantification over a paths E and A. Okay.

**(Refer Slide Time: 07:51)**




This is called CTL*. CTL*is made up of State formulae as well as Path formulae there is a mutually recursive definition because you see Alpha use a State formulae and State formulae use Alpha again. Okay. So CTL* is made up of State formulae and Path formulae. Let us look at certain examples. You can say E of Alpha and Alpha can be F of some Path formulae again which could be phi and phi could be P1.

So you know how you can get E F P1. Let me repeat so E F P1 is a valid State formula because a State formula-- see look at E F P1 it is of the form E of alpha, right. Where Alpha is F of some Alpha one, look at this Alpha one could be phi and phi could be P1 so that is how you get E F P1. What about A F A G P1 it is again a State formula because it is of the form A alpha were alpha is F of A G P1.

Look at this it can be F of Alpha one, so A G P1 should be obtained on some Alpha one but this is-- for this you use the fact that Alpha can be one the state formulas and here you can use A so you get A F A of Alpha were alpha is G P1 NOT. Okay and you how to get G P1. So this is a valid formula that can be obtained out of this syntax. This is the syntax of CTL* and you can derive this from the syntax.

What about A F G P2? See P2-- okay let us start from here. It is the State formula because it is of the form A of Alpha where alpha is F G P2 this says that F G P2 says that it is the property of a path it says that in this path at some point of time rather-- from some point of time P2 is always

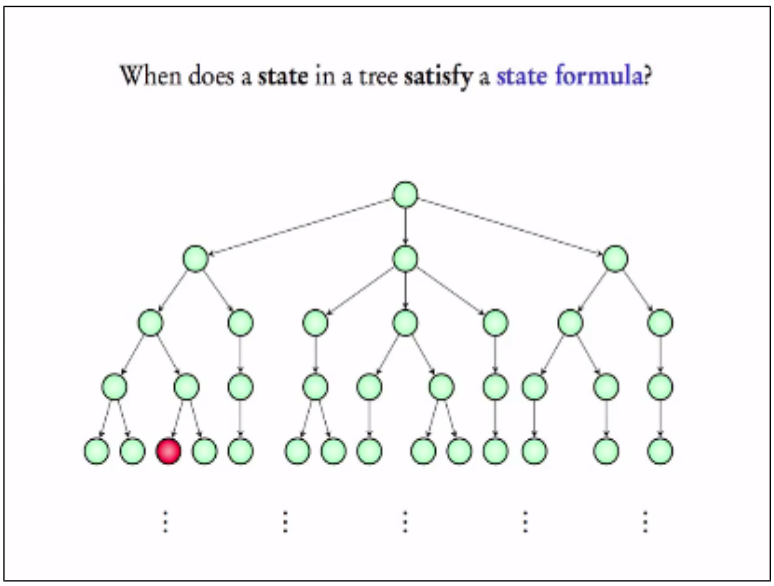
true. This is F G P2. From some point of time G P2 is a true and this is true over all paths so A of Alpha where Alpha is a Path formula F G P2.

How can you form F G P2 here? F of Alpha one where Alpha one could again B G of something and finally there something is a state formula were you have P2. So A F G P2 can be derived using these rules. These are just rules of deriving formulae. You can also say A P1. Let us see the meaning of this later but you can say A of alpha were alpha could be a State formula and the State formula is P1.

You can also do this A P1. Because A of Alpha—Alpha could be a State formula which could be E of Alpha and again Alpha could be State formula P1 okay, so these are all valid formulae of CTL*. This is giving us the syntax of CTL*. Any string that can be derived out of these rules is called a CTL* formula and it could either be a State formula or a Path formula. Let us now try to attach meanings to these strings.

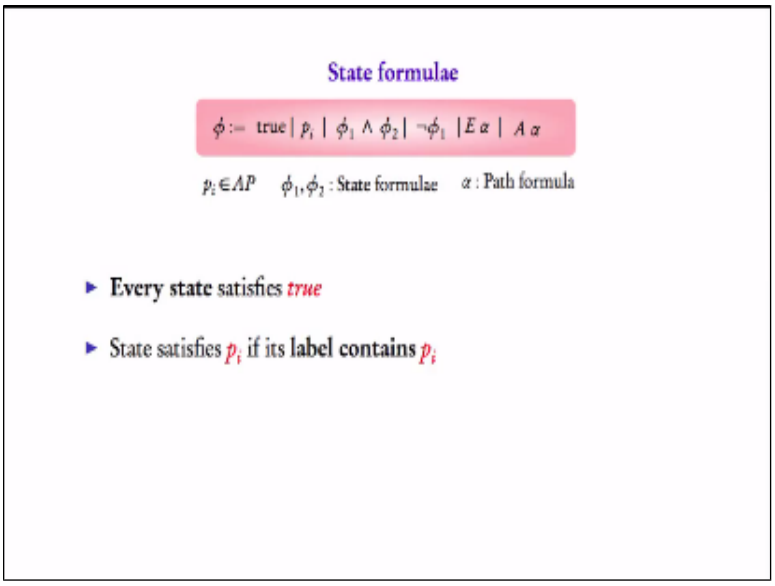
We saw certain examples and we saw what they mean in the last module. However, let us try to give a precise definition to each of these strings.

**(Refer Slide Time: 12:07)**



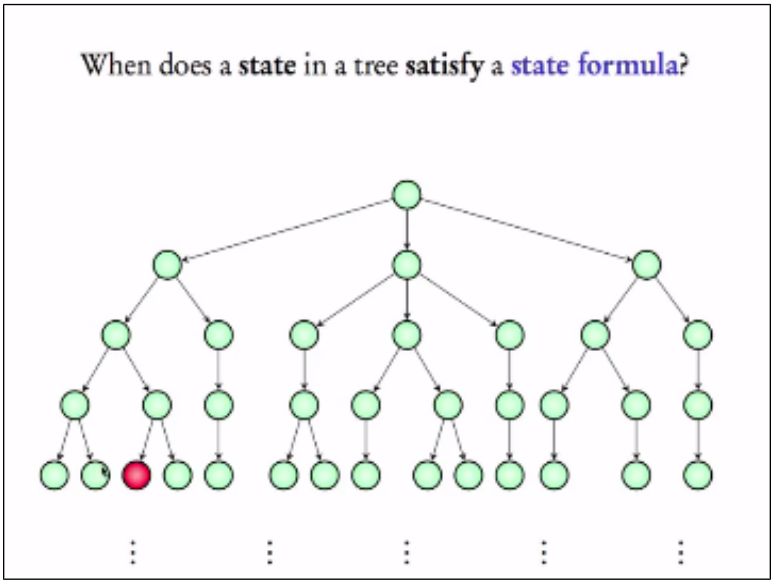

So we ask this question. State formulas are evaluated on states in the tree. So when does a state in the tree satisfies a State formula? This the question that we ask.

**(Refer Slide Time: 12:23)**



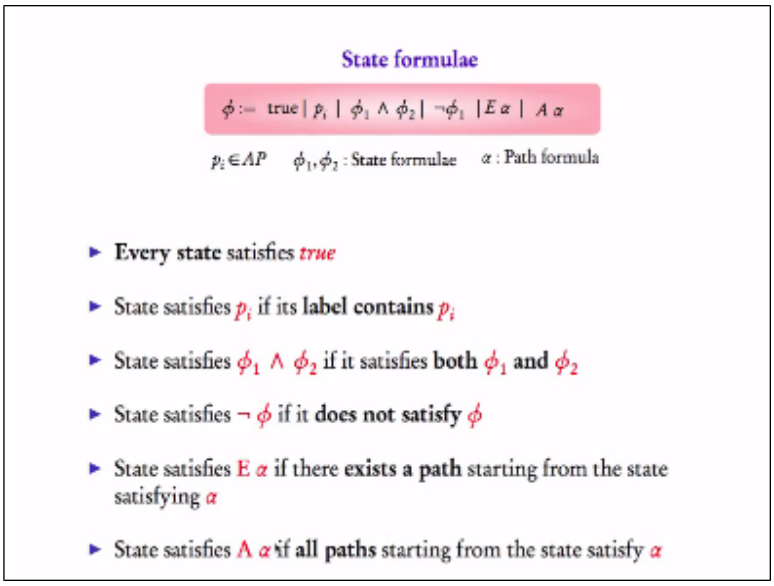

These are the State formulae. Every state in the tree satisfies true. A state satisfies Pi if its label contains Pi.

**(Refer Slide Time: 12:42)**




So this state will satisfy red. For example, if this state has P1, P2 and it satisfies P1 and it also satisfies P2.

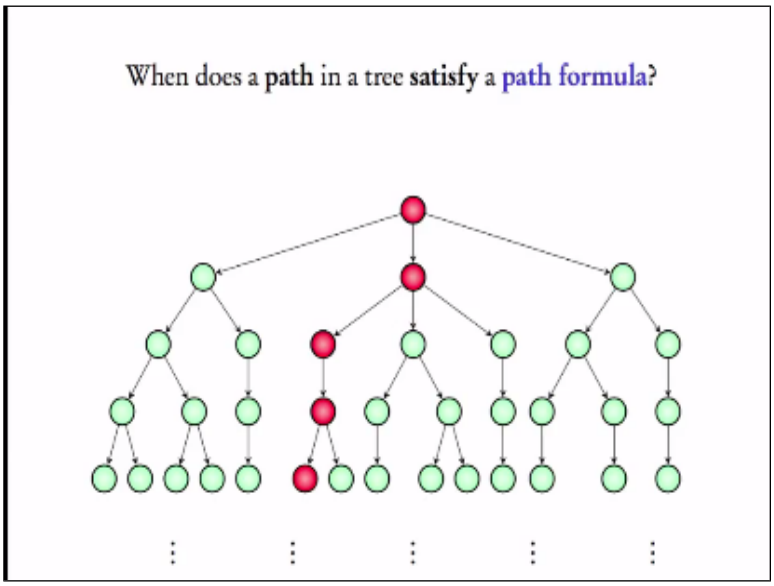
**(Refer Slide Time: 12:57)**




A state satisfies phi one and phi two if it satisfies both the state formulae phi one and phi two. Similarly, a state satisfies NOT of phi one if it does not satisfy the state formula phi one. What about this? A state satisfies E Alpha if there exists a path starting from the state which satisfies alpha I mean the path starting from the state should satisfy Alpha. We have seen this before. Similarly, a state satisfies A Alpha if all paths starting from that state satisfy Alpha.

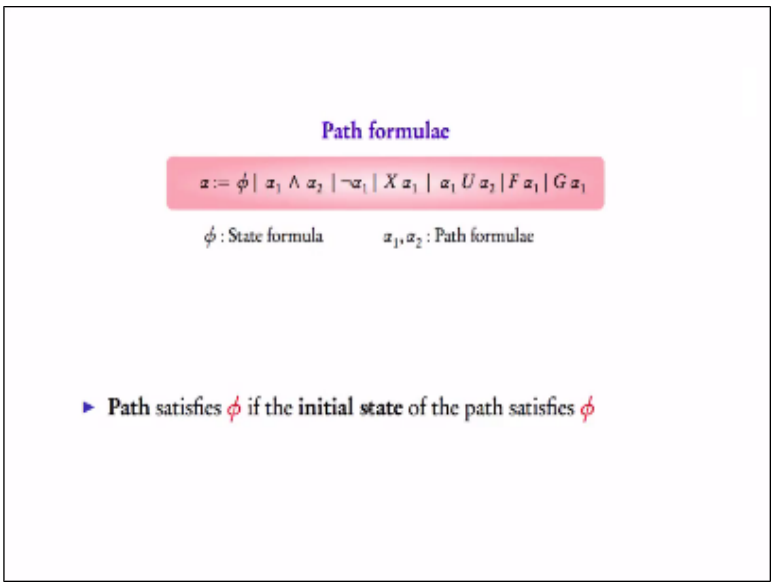
This could be any state in the tree not necessary the initial state. The state formula can be evaluated in any state of the tree, and this is the semantics this is the meaning of these formulae and this slide answers the question, when does a state in a tree satisfy a state formula.

**(Refer Slide Time: 14:10)**



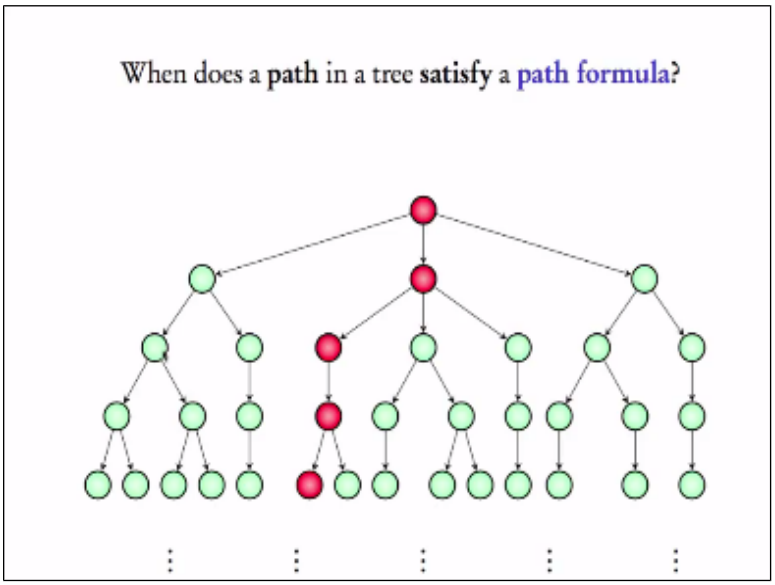
Now what about the Path formulae? When does a path in a tree satisfy a Path formula? This path could start from a any state I have just given this for illustration. You have a path starting from some state and when does this path satisfy a path formula?

**(Refer Slide Time: 14:33)**



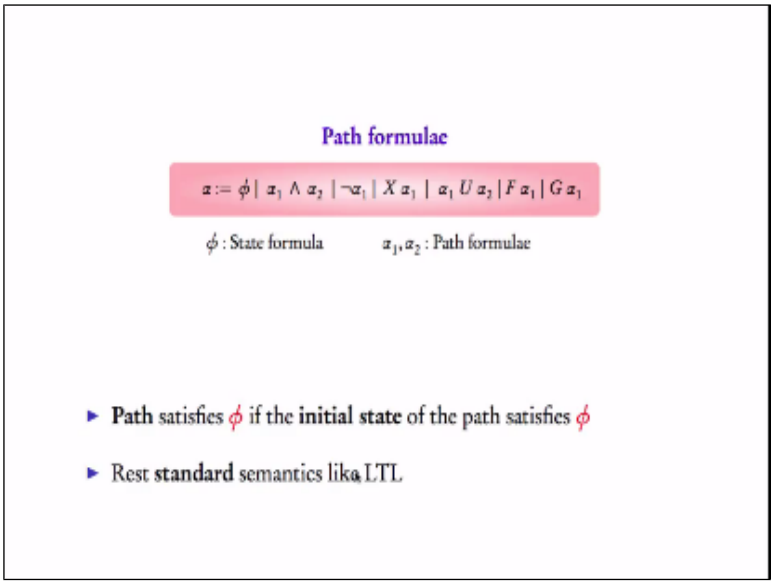

So these are the possible Path formulae. Most important is this base case. When does a path satisfy the State formula phi; path satisfies phi if the initial state of the path satisfies phi okay.

**(Refer Slide Time: 14:56)**



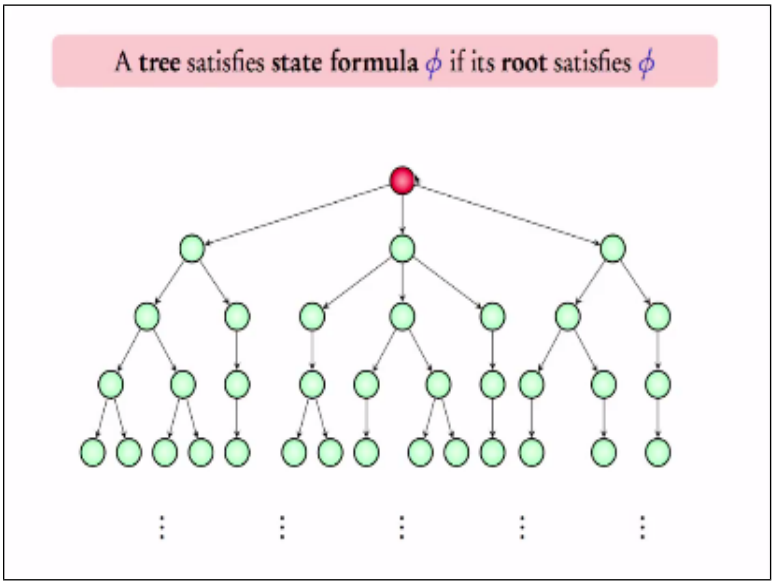

Suppose you have a path starting from this state say and you have a State formula P1 and P3 this path will satisfy it if this state satisfies P1 and P3. The State formula could also A X something and the path starting from this will satisfy the state formula if this state the initial state satisfies the state formula. The rest are standard.

**(Refer Slide Time: 15:26)**



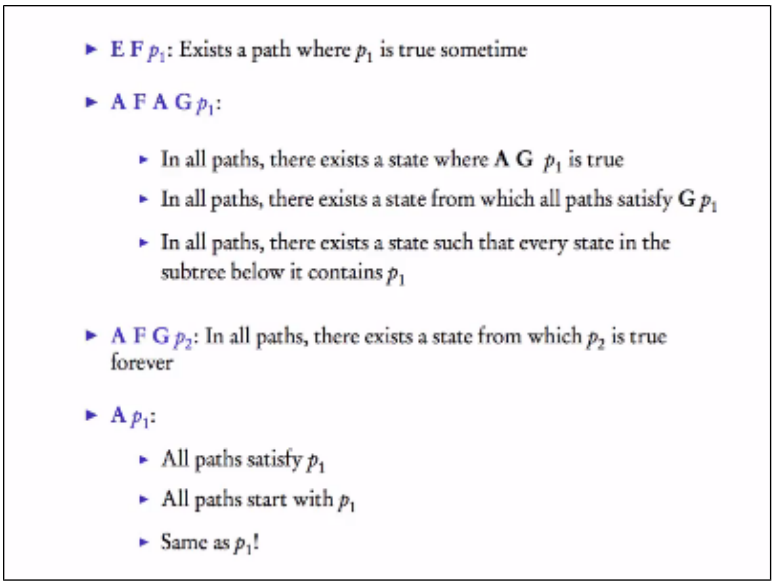

A Path satisfies Alpha one and Alpha two if it satisfies both of them. I mean it satisfies NOT of Alpha one if it does not satisfy Alpha one. For X U F and G its standard like LTL so I do not want to repeat.

**(Refer Slide Time: 15:48)**




Now, we saw well a state in a tree satisfies a State formula. When a path in a tree satisfies a Path formula and now when does a tree-- suppose you are given just the tree and the state formula when does the tree satisfy the state formula, we say that a tree satisfies State formula phi if its root satisfies phi. So this tree satisfies say A F P1 if from every path starting from the root F P1e is true.

**(Refer Slide Time: 16:37)**



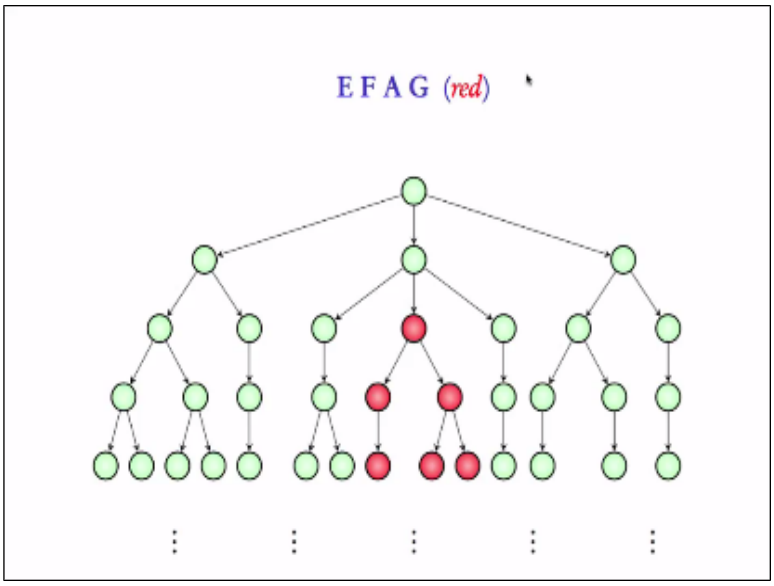

So let us see some more examples. E F P1 exists a path starting from the root – when it is not mentioned you can assume that the path starts from the root, exists a path where F P1 is true that means where P1 is true sometime. What about this? A F A G P1 this is a valid formula we solved how it is derived. Now what does it mean? In all paths, there exists a state where A G P1 is true. This is for A F, in all paths there exists a state where A G P1 is true.

In all paths starting from the root there exists a state where A G P1 is true. In all paths, there exists a state from which all paths satisfy G P1. Right, that is the meaning of A G P1. Now, this just means that in all paths there exists a state such that every state in the subtree below it contains P1. Okay. This is the meaning of A F A G in all paths there exists a state up to this A F such that every state in the subtree below it contains P1, this is for A G A G P1.

What about A F G P2? In all paths, there exists a state where G P2 is true that means P2 is true forever starting from that state. In all paths, there exists a state from which P2 is true forever. What about A P1? All paths satisfy P that means all paths start with the state labeled with P1. This is true as the state formula P1 itself. Because it just needs the root to satisfy P1. And if you evaluate the state formula P1 on the tree it means that you are evaluating it on the initial state.

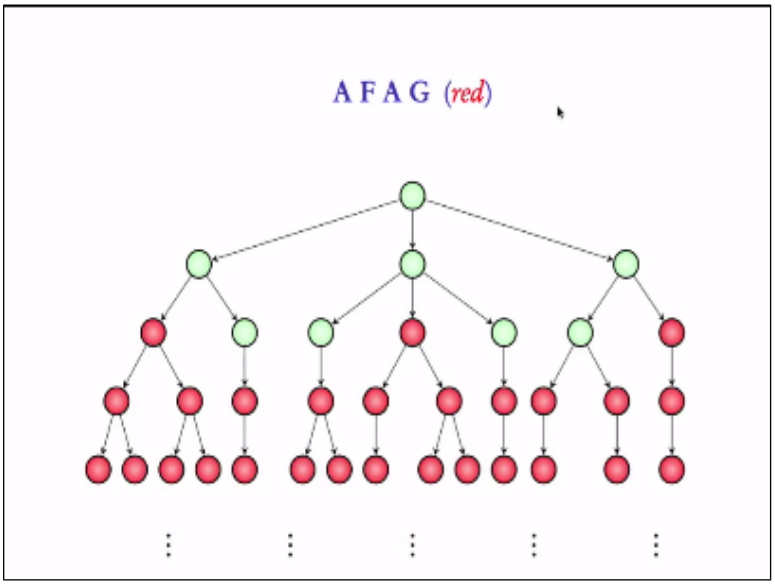
So A P1 is actually the same as P1. So these are some examples of the properties expressed using CTL*.

**(Refer Slide Time: 19:27)**



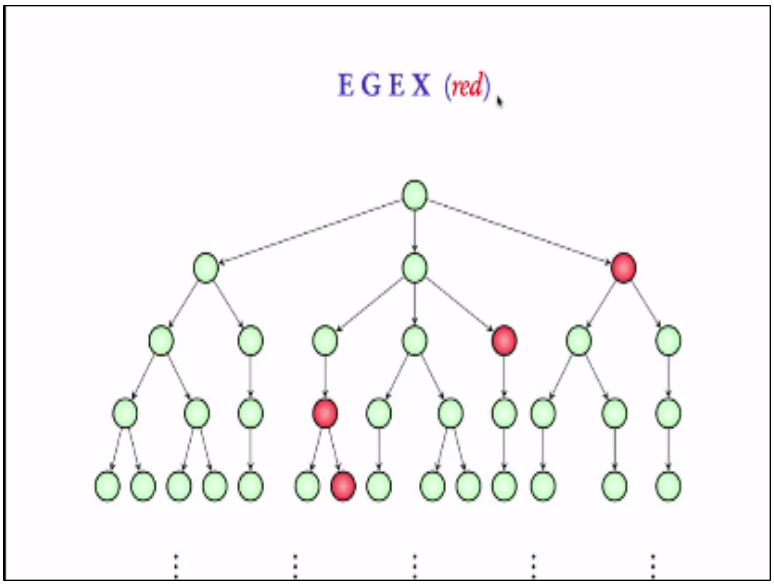

Let us now relook the examples of the previous module with this additional semantic defined E F A G exists a path that reaches a state, see F A G (red) so there exists a path where F of A G red is true. That means F of-- I mean that means there exists a state where A G red is true and this that state. A G red this is a State formula.

**(Refer Slide Time: 20:04)**



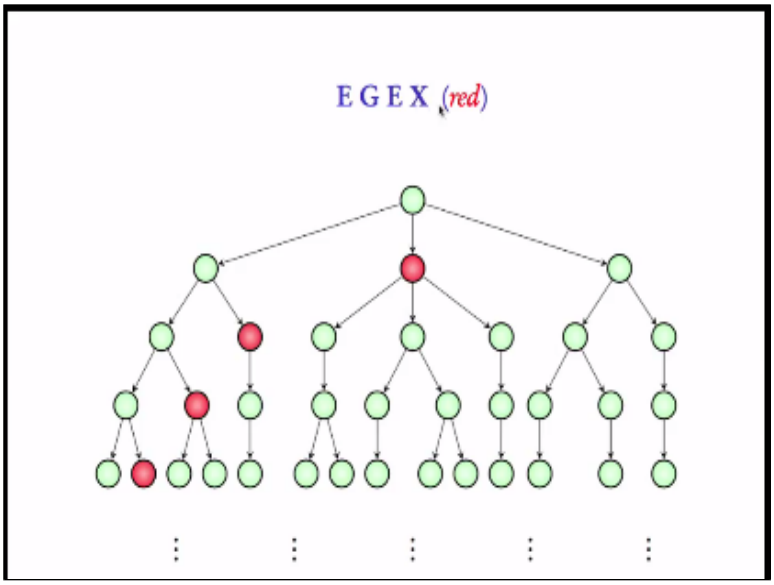

A F A G, we have seen this before so from all paths there exists a state that satisfies A G red.

**(Refer Slide Time: 20:13)**



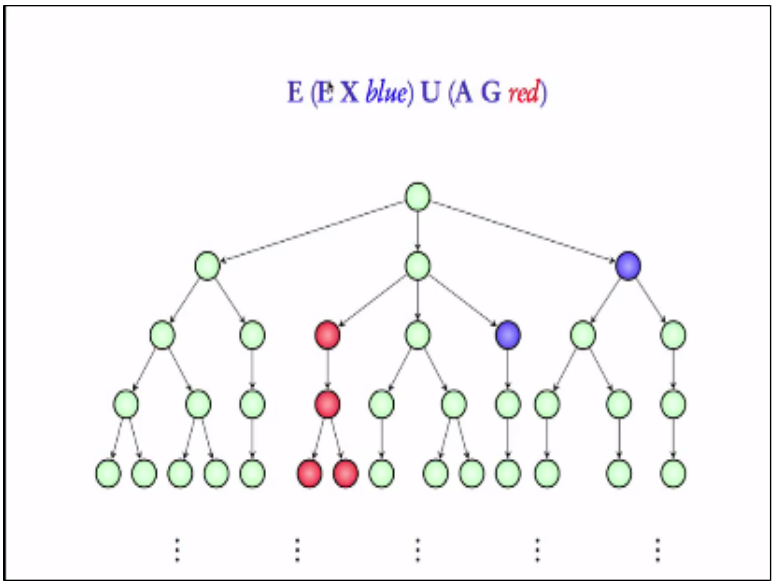

E G E X there exists a path such that every state satisfies the State formula E X red. So this satisfies E X red; this satisfies E X re; this satisfies E X red and so on. So E X red is a State formula. G E X red is a Path formula. And E G E X red is again a State formula.

**(Refer Slide Time: 20:40)**



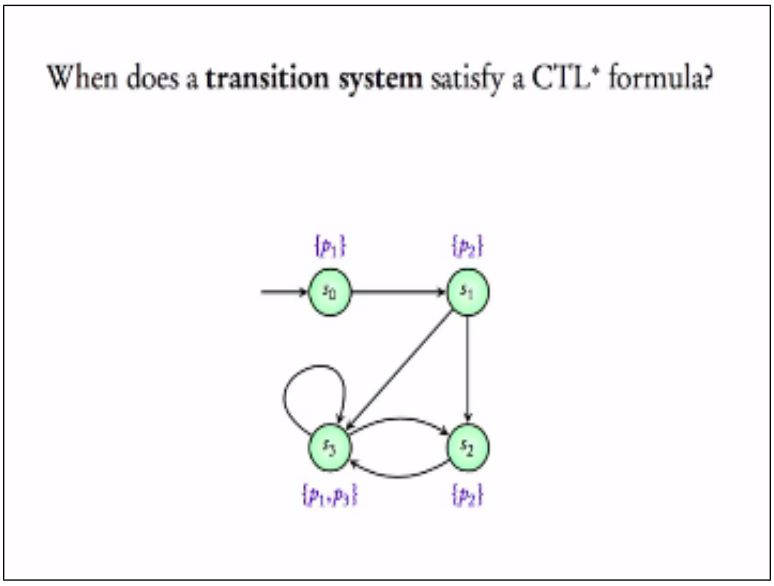

This is one more tree which satisfies E G E X (red).

**(Refer Slide Time: 20:47)**




Now this is see (E X blue) is a State formula. Similarly, (A G red) is a State formula. And this until this is a Path formula which is true on this path. This state satisfies (E X blue). This state satisfies (E X blue). This state satisfies (A G red) so this path satisfies (E X blue) until (A G red) and E of this is again a State formula which is true on this state. So there exists a path satisfying this until formula.

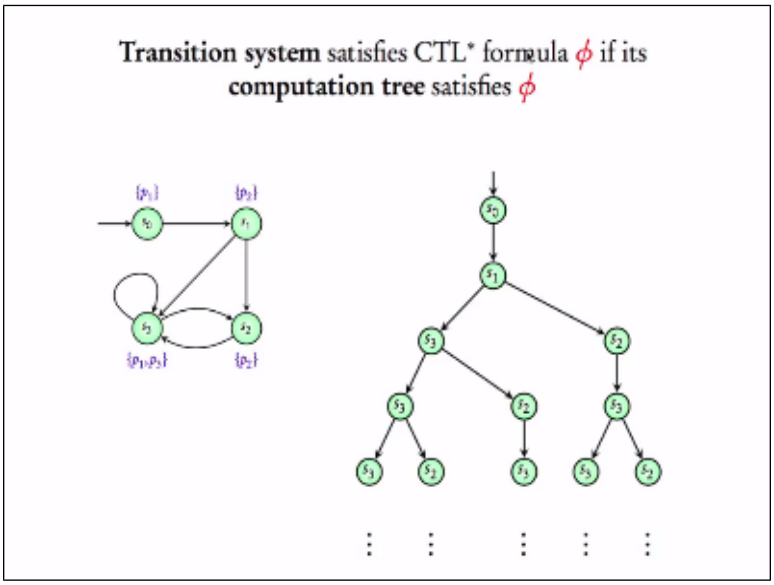
**(Refer Slide Time: 21:24)**




Now so far what we have seen is we have given you the logic CTL* made up of State and Path formulae. We saw when a tree is set to satisfy CTL* State formula. It is set to satisfy the state formula if the initial state I mean the root satisfies it. Finally, what we are interested in is a

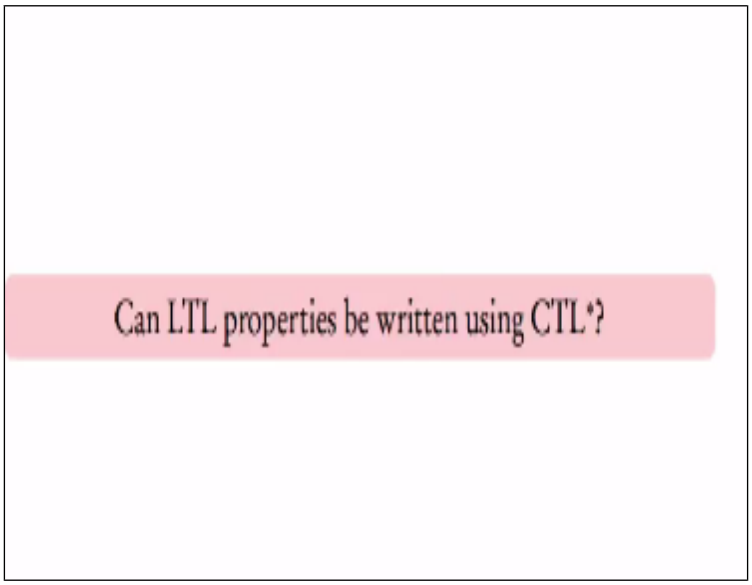
transition. Given a transition system and a CTL* formula when do we say that the transition system satisfies this CTL* formula?

**(Refer Slide Time: 22:05)**



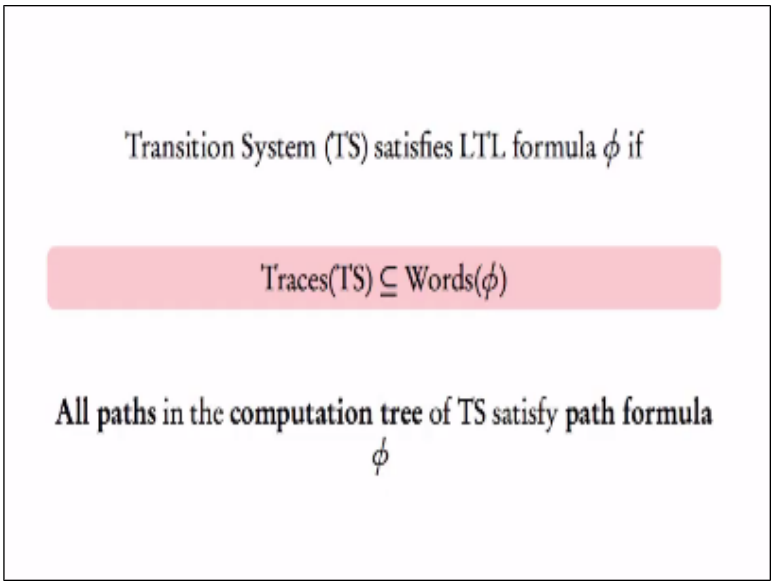

The Transition system satisfies CTL* formula if its Computation tree satisfies phi. That means you have to evaluate the CTL* formula on this initial state on this root and if the formula is true on this tree then this transition system is set to satisfy the CTL* property.

**(Refer Slide Time: 22:33)**



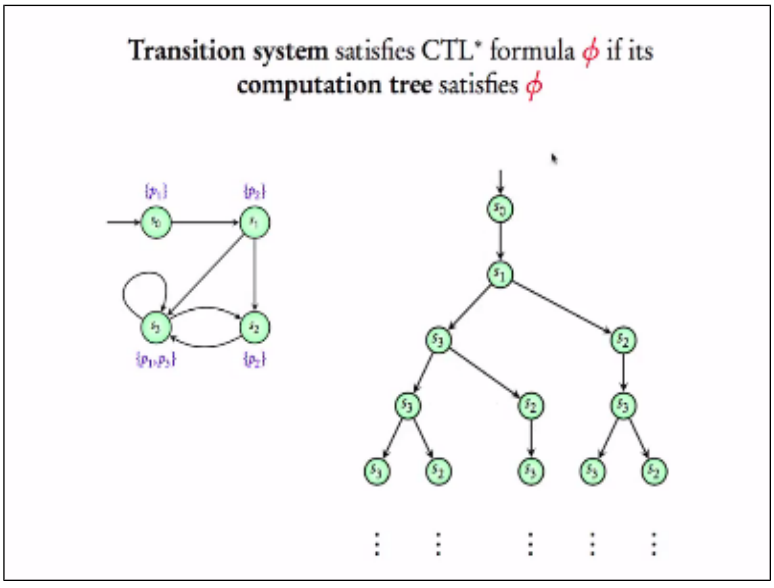

Let us now ask a different question. We saw property specification using LTL. Now we are seeing properties specification using CTL*. Can the property which can be expelled using LTL be written using CTL*? Let us see.

**(Refer Slide Time: 22:57)**



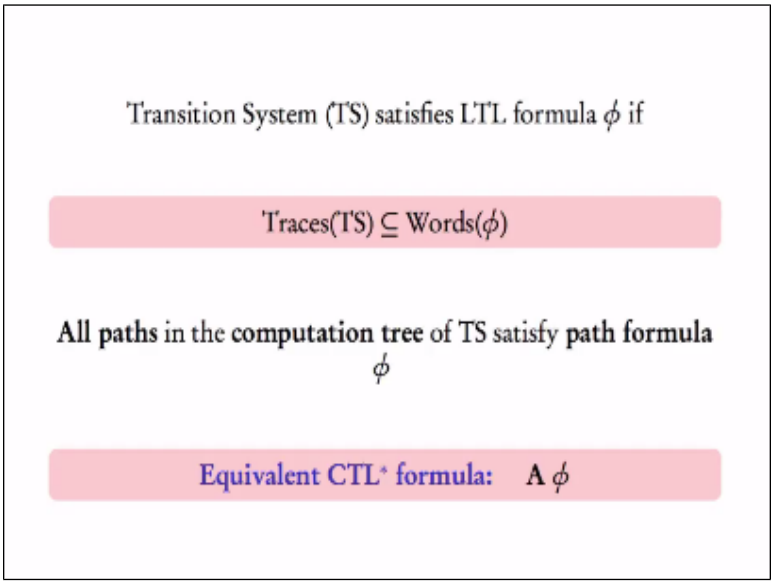

Recall that a Transition System satisfies the LTL formula if the traces of the transition system that is all paths satisfy the formula phi in other words the traces that contain inside words of phi. We have seen this many times before. So this LTL formula gives rise to set of words and these transition systems traces again a set of words this set should be contained in this set. That means all paths in the computation tree satisfy the path formula phi.

**(Refer Slide Time: 23:43)**



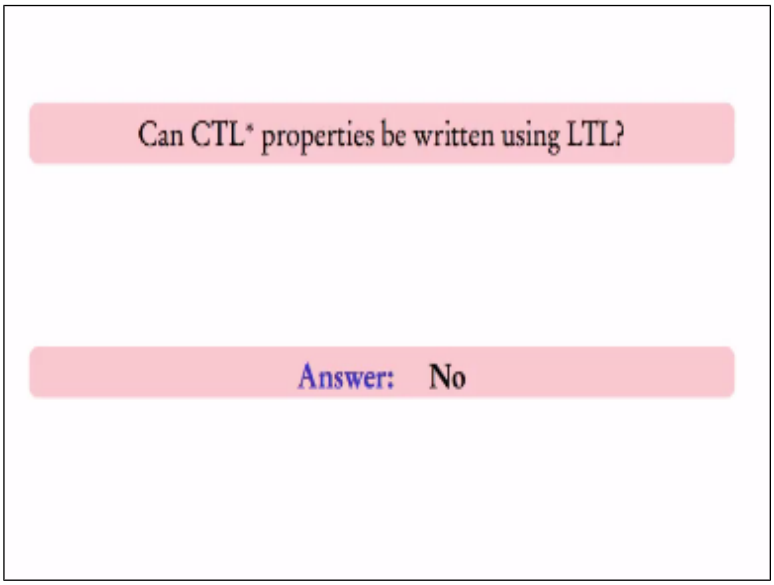

If you look at this transition sorry if you look at this computation tree paths in this I mean traces correspond to paths here-- mean paths of this transition system correspond to paths of this computation tree. And by seeing that every path satisfies the formula we are just seeing that every path in this computation tree satisfies the path formula phi.

**(Refer Slide Time: 24:12)**



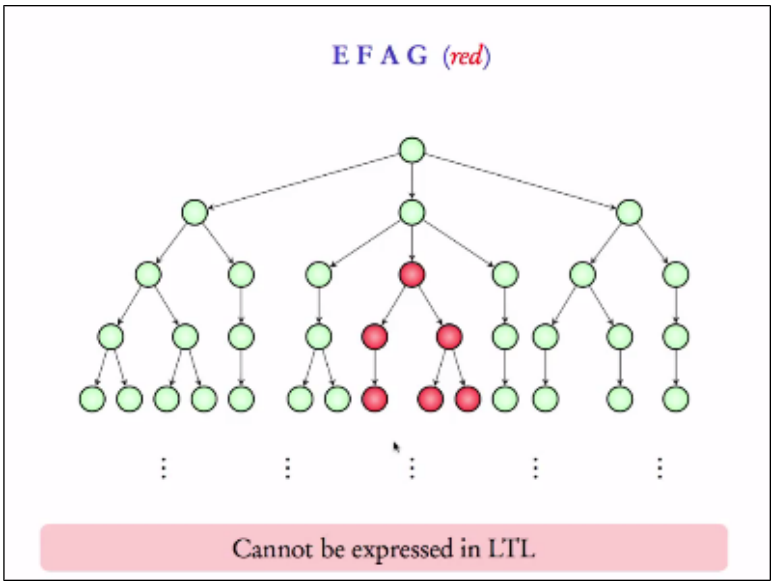

Hence the LTL formula phi is the same as the CTL* formula A phi. If you evaluate-- I mean in other words if the transition system satisfies the LTL formula phi the computation tree of the transition system will satisfy A phi. Similarly, if the computation tree satisfies A phi then the transition system will satisfy the LTL formula phi. So the answer is LTL properties can indeed be expressed using CTL*.

**(Refer Slide Time: 24:54)**



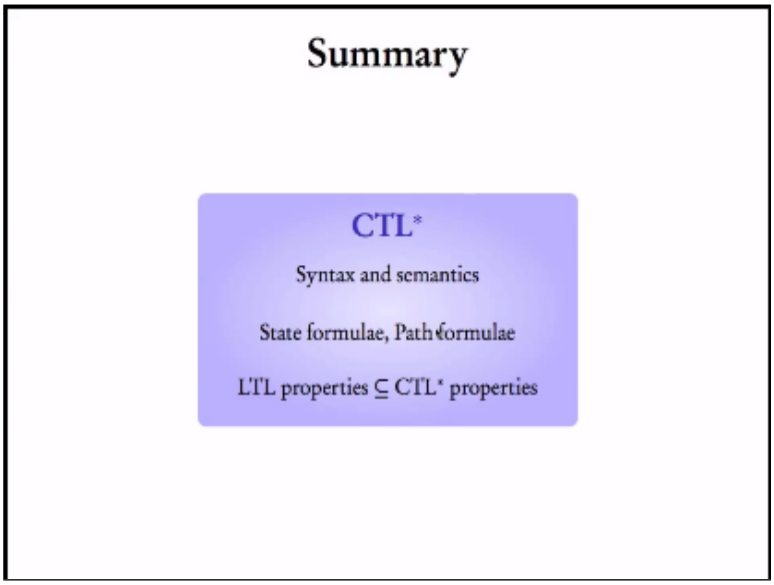

Now what about the reverse question? Can CTL* properties be written using LTL? As and if I want to talk something about the tree can I write it using LTL which deals only about paths as you might have guessed the answer is No.

**(Refer Slide Time: 25:16)**




Look at a property like this; you want to say that in the transition system there exists a path from which all possible continuations satisfy certain property. This kind of a mix cannot be expressed using LTL it needs CTL*. So this says that there exists a competition which reaches some state from which all possible extensions satisfy a certain property.

**(Refer Slide Time: 25:52)**




In this module, we have seen the logic CTL* which is made of Path and State formulae. You need to understand the distinction between this. Please go through the examples once again if you have not understood this. We have shown that LTL properties can be expressed using CTL* properties and in fact this is a strict containment in the sense that there are CTL* properties which cannot be expressed using LTL properties.