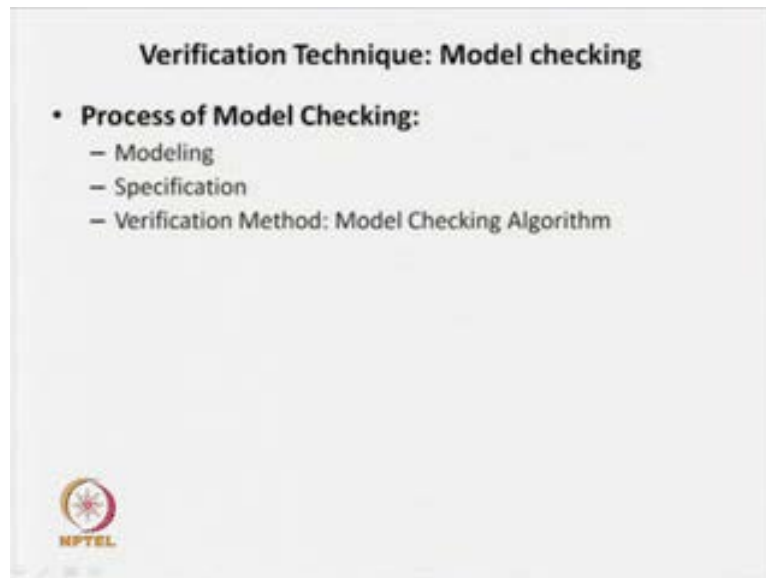


Design Verification and Test of Digital VLSI Designs
Prof. Dr. Santosh Biswas
Prof. Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Model - 5
Verification Techniques
Lecture - 2
Model Checking Algorithms

(Refer Slide Time: 00:27)



In last class, we have discuss something about your model checking; so what happens basically, it is a verification techniques and in this verification technique call model checking. We are having three components basically, what is your modeling of the system? Number 2 specification; basically we have to give the property, we have to give the property in some formal languages. So can you, so that we can use some formalism to verify those properties in the model, and the verification method that we are going to talk about your model checking. So in last class I have introduced the concept of model checking with the help of an example; where we have seen that the designing of the mutual exclusion protocol.

Now with this particular small example, what we have seen that, how to model a system? That means how to come up with the model of this particular system; then we have to seen, what are the properties that system for that mutual exclusion protocol should satisfy? We have note down those particular property like, safety property; liableness

property; then what we talk about non-blocking and knows three sequencing. Now these are the property or these are the requirements for our system; then what happens? Those particular specifications have to be captured in some formal way or in some formal language.

Already we have discussed about CTL, Computational Tree Logic and we have seen how they up to be define or express in your CTL. Then we have to seen model look into this particular specification; now we are going to check whether those specifications are true in this particular model or not. We have seen that the model that we have come up is not satisfying the lioness property, but we can modify it so that it can satisfy the liableness property also.

Now with this small example, we have seen or I have given in the (()), how model checking works? Now like that, when we are going for a bigger system, then will be having lot of steps; the steps has will be more and already we have seen that, basically the number of steps in our model will depends on the number of control signals that we have. It is basically exponential to the number of signals that, we have in the system which is equal to 2 to the power m.

So basically, when we have going to look for a bigger system and we are having a complex formula or complex specification. How to check for a correctness of those particular for a specification in the model, for that we need some algorithm; we need some mechanize method and today we are going to see, what are the algorithms? That we have for this particular model checking methods.

(Refer Slide Time: 03:06)

The slide is titled "Model Checking Algorithm". It contains the following text:

Given the model ' M ' and a CTL formula Φ as input.

Model checking algorithm provides all the states of model M which satisfy Φ .

Handwritten annotations include a double underline under the word "input" in the first line, a double underline under the symbol Φ in the second line, and a double underline under the symbol Φ in the third line. In the bottom left corner, there is a handwritten "M" and a handwritten Φ next to the NPTEL logo.

So we are going to talk about model checking algorithm today. So in last class, I have also slightly introduce about this particular model checking algorithm. So basically what we have in this particular model checking algorithm? We are having a model M . So that system model M , that will be having will come up with this particular system model and along with a one CTL formula ϕ that means, this is the property or the specification model system. Now how model checking algorithm is going to walk? So model checking algorithm provides all the states of model M , which satisfy ϕ . So basically you just say that, we have going to give a model M and will give a CTL formula ϕ and we are going to collect all the states; where this particular formula ϕ is true in the model M .

So basically, ultimately we are coming up with an labeling algorithm. So our basic aim is to find total an labeling algorithm; these labeling algorithm is going to level the states of this particular model with a formula; if it is true in the particular states and finally we are going to written those particular states, where this particular formula is true. So this is about the model checking algorithm that, we are going to look into it.

(Refer Slide Time: 04:18)


Labeling Algorithms

CTL model checking algorithm basically works by iteratively determining (i.e., labeling) states which satisfy a given CTL formula.

The basic input/output of labeling algorithm are as follows:

INPUT : A CTL model ' M ' = (S, \rightarrow, L) where S is the set of states, \rightarrow is the transition relation and L is the labeling function and a CTL formula Φ .

OUTPUT : The set of states of M which satisfy Φ .



So basically, what labeling algorithms? Now we can look into the, this way CTL model checking algorithm, basically works by iteratively determining. That is labeling states, which satisfy a given CTL formula; so here we are saying that, it is work by iteratively determining will see or iteratively we are having iterative method, which is going to label the states with the help of the formula; where it is true after that, after completion of this labeling algorithm it will written all the states, where the given formula is true; the basic input output to labeling algorithm are as follows.

(Refer Slide Time: 05:27)

Labeling Algorithm ϕ

- The adequate set of temporal operators for CTL is AF, EU and EX.
- First, we write the given formula Φ in terms of the connectives AF, EU and EX along with other logical connectives and truth value T .
- Suppose ψ is a subformula of Φ and states satisfying all the immediate subformulas of ψ have already been labeled.



So what is the input? So we are having a CTL model M and we know that, this model is nothing but set of states one transition relation and the labeling function. So these are the input to the model and where S is the set of states, that arrow is the transition relation and L is the labeling functional; along with that, we are going to give a CTL formula ϕ . So what will be the output? The set of states of M which satisfy this particular ϕ .

So what is the labeling algorithm? Now say we have seen that, what are the operators that we have temporal operators; we have in our CTL and here basically, temporal operator called next state future, global and until; and these particular 4 temporal operators will come up with the path quantifier a and e , so all together we are getting 8 different combination.

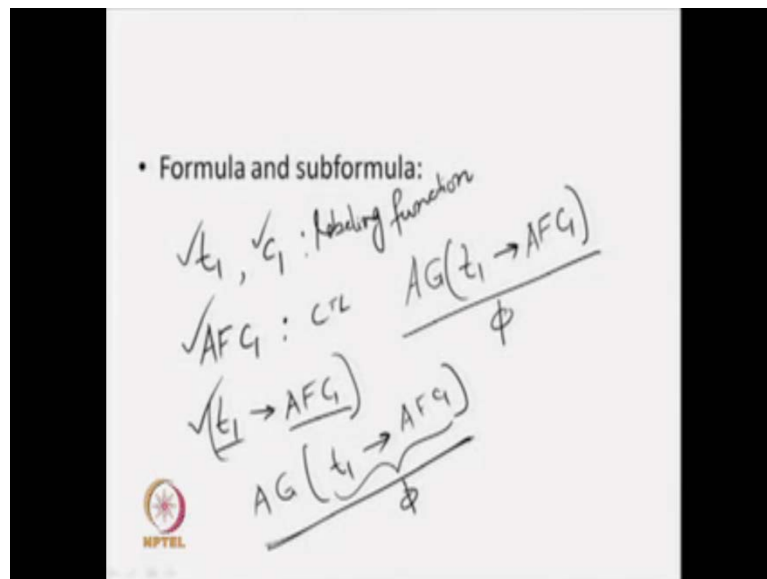
So we need the method for those 8 different combinations or 8 different CTL operator, but all that we have seen that out of those aid, we need only 3 which are going to form the adequate set of operator. So if we know the method of key operators, then we can go for all those particular aids; so we know that adequate set of operators, we are going to take one particular adequate set of operator which includes AF, EU and EX; that means in alpha infuse, they are exist a path with until and they are exist a part next state.

So will just say that, since this is the adequate set of operator; we are going to look for a method, these three particular operators and once I get the method for these three operators, then other can be expose with the help of these three operator. So when we are getting a particular formula ϕ than first of all what we are going to do we are going to express this particular formula in terms of the connective AF, EU and EX because we are going to have method for these three operators, along with the logical connectives and truth values true because we are having truth values true one is true, which is represented top and another one is bottom, which is represented by faults; which is represented by bottom.

So we are going to look for the all logical connectives and that truth values true top and true which is top and faults which is bottom. So if any ϕ we are getting any CTL formula ϕ , we are getting first we are going to express this particular CTL formula with the help of these three operators because we know the equivalence; we are having some equivalence relation in CTL. Now suppose ψ is a sub formula of ϕ , know we are going to look for some sub formula.

So if ψ is a sub formula of ϕ and states satisfying all the immediate sub formulas of ψ have already been labeled. Now what does it mean immediate sub formula means, all the sub formulas that we have. So basically what will happen? When we are going to look for a particular formula, first of all we know the components and we should know the truth values of those particular components. These are basically sub formulas; once we know the truth values of the formulas or sub formulas, then we can look for a truth value of the main formula.

(Refer Slide Time: 08:17)



For example, I can say that a formula say in our mutual exclusion formula; mutual exclusion your protocol example what happens? We have seen the liveness properties like that in $AG t_1$ implies $AF C_1$; so this is the property say CTL property in $AG t_1$ implies $AF C_1$; so say that in all part globally if t_1 is true, it implies that in all part increases C_1 will be true. Now in this particular case, you see that when we are going to look for this particular formula ϕ , then we have come up with the orders of formulas of this particular ϕ .

So here t_1 and t_2 are your atomic proposition so they will be CTL formula. So we are going to check say that, these are two CTL formulas; so one I am having t_1 and c_1 since these are your atomic proposition. So we are having that particular labeling function we know that, we have the labeling function and with the help of labeling function, we level the step; where the atomic propositions are true.

So we know the truth values of these two atomic propositions; we know model and in we know in which step these are true. Now once we know that, these two are CTL formula; then will find that $AF C 1$, $AF C 1$ is also a CTL formula. Next what will happen? We have to level all this stage with this $AF C 1$. Once we know the truth values of $AF C 1$ in all the states than only we can go for the next level; so that is why I am saying that, first we have to look for all sub formulas, once particular sub formula is labeled than we can do for the next one.

So once it is a CTL formula now with the help of a algorithm you look for the states where this particular formula is true then by looking into the main formula ϕ than we are going to get this formula $t 1$ implies $AF C 1$. So again this CTL formula it is having two components $AF C 1$ and $t 1$; now again with the help of our labeling algorithm, we are going to labeled the step with this particular formula $t 1$ implies $AF C 1$. So this is also sub formula of the given formula.

Now all the steps will be labeled this particular sub formula; once we know the label of this particular sub formula, than we go for the next level next sub formula; which is your AG implies $t 1$ implies $AF C 1$. Now we know the level of this particular formula; so you know the step where this particular formula is true. So once we know this thing then we can go for this particular our given formula ϕ .

So that is why I saying that, we must know the truth values of our sub formulas that means, we should know the step where the particular sub formula is true and where this particular sub formula is true. We are going to level those particular steps with the help of this particular sub formula, with the help of our labeling algorithm. So this is the way that we are going to look into it; so when we are going look for this particular ϕ , we should know the label or we should know the truth values of all those particular sub formulas $t 1$, $C 1$ and $AF C 1$ and $t 1$ implies $AF C 1$. So once we know this thing, then we can go for the complete given formula.

(Refer Slide Time: 11:55)

CTL Model Checking

Function $SAT(\Phi)$
 * determines the set of states satisfying Φ *

Begin
 Case
 Φ is \neg : return $S - S$ → $S - S = \emptyset$
 Φ is \perp : return \emptyset
 Φ is atomic: return $\{s \in S \mid \phi \in L(s)\}$
 Φ is $\neg \phi_1$: return $S - SAT(\phi_1)$
 Φ is $\phi_1 \wedge \phi_2$: return $SAT(\phi_1) \cap SAT(\phi_2)$
 Φ is $\phi_1 \vee \phi_2$: return $SAT(\phi_1) \cup SAT(\phi_2)$
 Φ is $\phi_1 \rightarrow \phi_2$: return $SAT(\neg \phi_1 \vee \phi_2)$
 Φ is $AX \phi_1$: return $SAT(\neg EX \neg \phi_1)$
 Φ is $EX \phi_1$: return $SAT_{EX}(\phi_1)$
 Φ is $A(\phi_1 U \phi_2)$: return $SAT(\neg (K \{ \phi_2 \cup (\neg \phi_1 \wedge \neg \phi_2) \} \vee XG \neg \phi_1))$
 Φ is $K(\phi_1 U \phi_2)$: return $SAT_{EX}(\phi_1, \phi_2)$
 Φ is $KF \phi_1$: return $SAT(K(T U \phi_1))$
 Φ is $KG(\phi_1)$: return $SAT(K(T U \phi_1))$
 Φ is $AF \phi_1$: return $SAT_{AF}(\phi_1)$
 Φ is $AG \phi_1$: return $SAT(\neg AF \neg \phi_1)$
 end case
 end function

$A \rightarrow B = \neg A \vee B$
 Φ
 M
 $SAT(M, \Phi)$
 P
 EU

NPTEL

Now we are going to see what will be the algorithms to check for such type of formulas. Now you just see that, we are in this particular model labeling algorithm what happens? We are going to give a formula ϕ and we are going to get a model M . So we are going to define a satisfaction function, which is on the function said and given input formula is ϕ and some model is by default or we can define that it is said can be define like that, set satisfy will be in the model M and the given formula ϕ .

Now in this particular case, now as further semantics of this particular CTL formula, what happens? We know that all steps will be labeled with your truth value true and none of the step will be labeled with the truth value false. So that is why if ϕ is true, then it is going to written the complete step space S because true is true in everywhere. If the ϕ given formula ϕ is your false bottom, then it will enter the non-step; so this is basically now it is true. So it is going to return the non-step because the truth value false is false everywhere; so now it will not labeled with this particular fault.

So if my given formula is your ϕ than, which is going to return on the non-step. Now if ϕ is atomic, now we are giving some atomic ϕ is equal to say some atomic variable P ; than what will happen? Then it will return the steps, those particular states where that states is a member of this particular labeling function of this particular step because we know that all states will be labeled by your atomic proposition.

So with the help of labeling function we are labeling the states so if my given formula ϕ is as an your atomic proposition if it is atomic so it is going to written all such type of states which are a member of this particular labeling function for that particular state S . Now if my given formula ϕ is negation of say ϕ_1 ; so it is a I am giving a formula ϕ which is negation of ϕ_1 . So you just see that I am saying negation of ϕ_1 so when we are going to look for this particular formula.

So ϕ_1 is a sub formula of this particular given formula; so we must know the label of ϕ_1 in each and every step. So in my given formula is negation of some CTL formula, then what it is going to returns? It is going to returns the states, where this particular ϕ_1 is not true; that means first we are going to have the label of the states, where ϕ_1 is true than the remaining state will be not of ϕ_1 so, it is going to written as minus set of ϕ_1 .

So this set is where to the ϕ_1 is going to give me the steps, where the given formula is true; so if negation of ϕ_1 it will be S , the total steps plus minus satisfiability of ϕ_1 . So you just see that, if not of ϕ_1 you can said that, it is having a sub formula ϕ_1 so first we know the labeling of ϕ_1 ; then only we can go for the knot of ϕ_1 similarly, ϕ is your ϕ_1 and ϕ_2 . This is the logical connective than what happen? It is going to return the step; first we know the labeling of ϕ_1 and ϕ_2 . So it is going to written two state, two sets; so the inter section of these two states will be the states where this particular formula ϕ_1 and ϕ_2 is true. So satisfy will be ϕ_1 is going to a set where the formula ϕ_1 is true so this is one state I consider the in this particular state; ϕ_1 is true and satisfy with the ϕ_2 is going to give me another set. It is going to say that, where this formula ϕ_2 is true.

Now for this particular ϕ_1 and ϕ_2 , that ϕ_1 and ϕ_2 must be true, in both are true in that particular states; so intersection will give me this particular state. So it is going to written this particular intersection point so similarly, ϕ_1 and ϕ_2 , ϕ_1 or ϕ_2 it will union of these two states; again we know that, if it is your ϕ_1 implies ϕ_2 , then written knot of ϕ_1 and ϕ_2 because this is equivalent. We know that ϕ_1 implies ϕ_2 is your equivalent to your knot of ϕ_1 or not of ϕ_2 ; that is why, let us you are going to look for the satisfaction of the satisfiability of this things and ultimately we are going to get this particular state.

Now we know this particular formula that $AX \phi_1$ is equivalent to knot of EX knot of ϕ_1 ; so you have seen this particular equivalent. So if my given formula is $AX \phi_1$, then we are going to look little written going to look for this things, satisfied with a knot of EX knot of ϕ_1 ; that means, first we have to express this particular AX in terms of EX because we are going to look for the minimum set up operator, which involves EX; another operator which is going to have EU and third one is your AX. So this is the minimum set of operators that we are going to look for it; so AX will be represented with the help of EX, now it is EX ϕ_1 .

Now say I am going to say that, there are exist a part next state ϕ_1 . Now what happens? For that it is going to written satisfiability with this particular EX ϕ_1 . Now you just see that in other cases, we are expressing it by some other things; now for EX, now this is a operator, now this is a member of my minimal set of operators; this is the minimal set or difficult set of operators that means, I need a method for EX now. We look for this particular EX. Similarly, if it is your A ϕ_1 until ϕ_2 , then we have discuss while discussing about the equivalence of CTL; that can be represent that, that A ϕ_1 until ϕ_2 can be expressed with the help of EU or EG. So this is the equivalent formula so when it is A ϕ_1 until ϕ_2 , then we are going to look for the satisfiability of this particular formula.

Now when E ϕ_1 until ϕ_2 , then when my input formula ϕ is your E ϕ_1 until ϕ_2 ; then we are going to return satisfiability EU ϕ_1, ϕ_2 ; that means we need a method for this particular operator EU (()), it is in my minimal set of operators. Now when my given ϕ is your EF ϕ_1 , then we are going to look for ET until ϕ_1 true until ϕ_1 because we know that futures can be expressed with the help of your, your until operator; so if it is your EF, then we need this particular each true until ϕ_1 similarly, if it is your AG EG, then what will happen? Is E ϕ_1 satisfiability of E true until ϕ_1 .

So AF so we need a satisfiability function like that AF is your satisfiability AF ϕ_1 . So since it is a member of my particular minimal set of operators so I need a method for AF. So this is another method and we need one method for set EU; similarly, as we can be express with the help of this equivalence, knot of VF knot of ϕ_1 .

Now any kind of formula I am giving as an input to this particular function set and it is giving to give me the set of states, where this for given formula is ϕ , given formula is

true for that since this is the minimal set of operator. So we need the method for this particular three operator EX, EU and F; and in this particular tree position, we are going to call the appropriate method.

So what we have seen that, whatever phi that we are having so if you (()) CTL formulas, than we are going to call the appropriate sub formula will convert it to the appropriate sub formula and we are going to get the set of states, were this particular given formula is true and since this is the minimal set of operators. So we need the method for set of EX, set of EU and set of AF.

(Refer Slide Time: 20:37)

The slide is titled "CTL Model Checking". It contains two main bullet points:

- Atomic proposition
 - p : label state s with p if $p \in L(s)$
- Logical connectives
 - $p \wedge q$: label s with $p \wedge q$ if s is already labeled with p and q

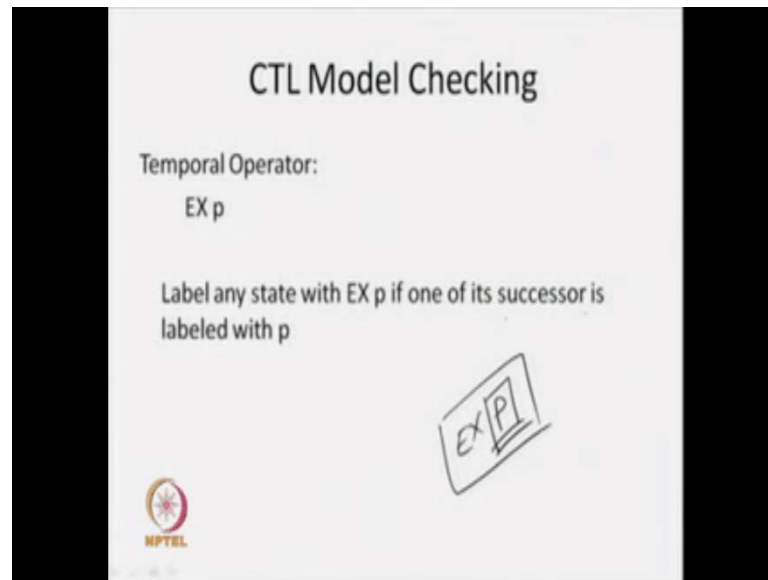
At the bottom left of the slide is the NPTEL logo.

Now we are going to discuss about these three procedures, how what will be the algorithms look likes for this particular EX, EU and AF operators. So it is, if we are having an atomic proposition p , than it is very we are going to label state s with p ; if p is a member of $L s$ because you just see that, we are having these things. If it is atomic, then we are going to written all the states where it is a member of this particular labeling function. So we are going to say that, if given formula is your atomic proposition; than we are going to label the state s with p ; if it is the member p belongs to yours $L s$ labeling functions of that particular state s .

Now similarly, if we are having any logical connectives say p and q , than label s with p and q ; if s is already labeled with p and q because we have seen this particular thing, if it is your p phi 1 and phi 2. Then we are going to look for the inter section of these two

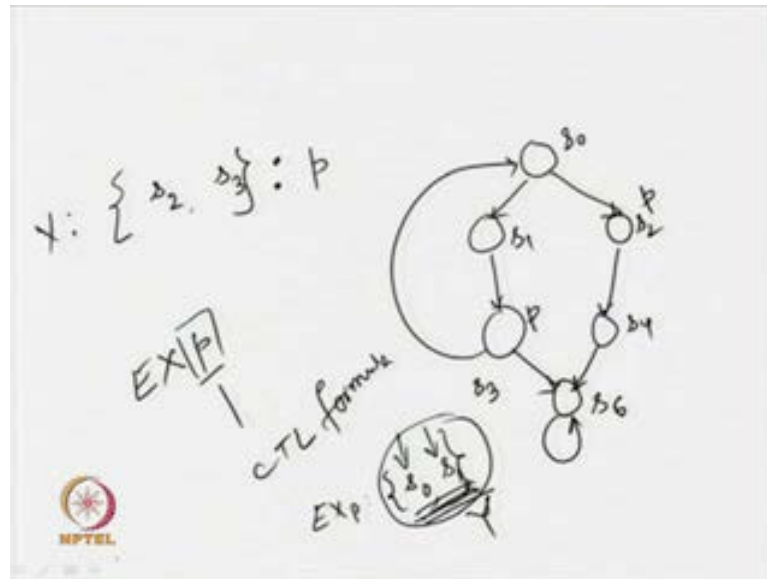
states so what will happen? If it is we are going to look for this particular p and q , then we are going to look for the states where both p and q are true. So if s is already labeled with your p and it is also labeled with q , then we can label this particular state s with p and q . So like that we can go for any other logical connectives.

(Refer Slide Time: 22:38)



Now temporal operator, we are going to look for say EX ; this is one of the temporal operators of our minimal set of operators or the adequate set of operators. So in this case it is there exist a part in next state p is true; so label any state with $EX p$, if one of its successor is labeled with p . So label any state with $EX p$ if one of its successor is labeled with p . So we are going to see the entire states place so in case of $EX p$; so p is a sub formula first we know the labeling of this particular p ; if we know these thing, then we can go for this particular given formula. So label any state with $EX p$, if one of its successors is labeled with p .

(Refer Slide Time: 22:39)



So say that one example, so this is a model you can say this is the cupcake structure so these are the set of states so we are having from s_0 to s_6 ; $s_0, s_1, s_2, s_3, s_4, s_5, s_6$. Now see that with the help of labeling function, this s_3 is labeled with p and say s_3 labeled with 3 that means, we are going to get a set where s_2 and s_3 . So in this set of states that p is true because these are labeled with this particular p , p may be atomic proposition or it may be any sub problem because when you go for $EX p$, we must know the level of p . So p may be any CTL formula.

Now what we are saying that, label any state with $EX p$; if one of its successor is labeled with p so in this particular case if you see these things. Now if you come to s_0 , then it is having two successors s_1 and s_2 ; so one of the successor is labeled with your p so it is a member of this particular states. So for $EX p$ what will happen? We will get that s_0 will come into these things.

When we come to s_1 , we will find that it is having one successor and this particular state s_3 is having this marked with labeled with p ; so we can say that s_1 will be labeled with your $EX p$. Now when you come to your s_2 it is having one successor s_4 , which is not labeled with p ; so s_2 will not be labeled with $EX p$. When we come to your s_5, s_3 then what will happen? You just see that, it is having two successors; one is s_6 and second one is your s_0 . So we will see both the successor none of the successor is labeled with your p so s_5 will not be labeled with your $EX p$.

When you compare s_1 , s_4 it is having one succor s_6 and it is not labeled with p so it will not be labeled with your $EX\ p$. When we come for s_6 , it is having successor s_6 itself; it is not labeled with p so it will not be labeled with $EX\ p$. So basically we are getting these two steps s_0 and s_1 ; who had $EX\ p$ is true. So my that satisfy will the algorithm or my that labeling algorithm will written this particular set of state s_0 and s_1 ; and it will say that in this two state, my $EX\ p$ is true.

(Refer Slide Time: 25:37)

CTL Model Checking

```

Function SATEX(p)
/* determines the set of states satisfying EXp */
local var X,Y
begin
  X := SAT(p)
  Y := {s0 ∈ S | s0 → s1 for some s1 ∈ X}
  return Y
end
  
```

EX p
X = SAT(p)
s₀ ∈ S

NPTEL

Now this is the algorithm you just see that, what is the function? It is function EX, satisfy with the SAT EX p that means, find out the set of states where the formula EX p is true. So we are going to look for the formula EX p determines the set of state satisfying EX p. Now here we are taking two local variable X and Y; now what is X? X first we are going to say that, it is calling this particular functions satisfiability with the sub formula. So it is going to returning the set of state, where this particular p is true; so like that if you are going to look into this things so this is the set X s_2 and s_3 .

Now what we are going to do? We are going to collect all those particular step s_0 ; so it is some set of s_0 , which belongs to that particular set of states of my model; in such a way, that there should be a transition from s_0 to s_1 , for some s_1 belongs to X. So in X what happens? We are having in all the states, where p is true. So now we are going to collect all such type of state s_0 where from this particular state s_0 ; we are having a

transition to the member of this particular state X . That means, we are going to get some of these states, where in next state this particular formula p is true.

Then after looking into it, then we are going to return this particular set of state Y ; so basically these particular states are going to give me a set of states, where the formula $EX p$ is true. So this is a very simple algorithm so giving the formula and if we know what is the sub formula p ; so we must know the labeling of this particular sub formula p . So we are first going to collect those particular states and say that this is my X because we are going now calling this particular satisfiability function with the p only or $EX p$. So once you get this thing, then now we can collect the order state where that $EX p$ is true. So we are going to look for all such type of transition, where s_1 must be a member of this particular set X and what is the set of this set X ? This is the set, where the formula p is true; so this is a very simple method.

So like that in previous example here we can say that first x is your s_2 and s_3 . Now from here, from those particular state we are going to look all the predecessor states; so it is going to give me s_0 , when it is your s_3 ; then from here we are going to look all the predecessor where it is going to give me s_1 . So these are the two states, where this particular formula $EX p$ is true and this is the Y , that we are having set Y that we have defined as a local variable.

(Refer Slide Time: 28:36)

CTL Model Checking

Temporal Operator:
 $AF p$

- If any state s is labeled with p , label it with $AF p$
- Repeat: label any state with $AF p$ if all successor states are labeled with $AF p$ until there is no change.

$AF p \equiv p \vee AX AF p$

The slide includes a diagram of a state transition graph with three nodes. The top node is labeled $AF p$. It has two children, both labeled $AF p$. The bottom-right child has two children of its own, both labeled MC . A box highlights the equation $AF p \equiv p \vee AX AF p$. The NPTEL logo is visible in the bottom left corner.

So next operator is your $AF\ p$, what is $AF\ p$? Basically in all part infuse of p must be true; so wherever you go we must get an states infuse, where p is true. Now how we are going to get an algorithm for this particular operator? You just see that, we are saying that if any state s is labeled with p , label it with $AF\ p$. Now say if we are going to look for $AF\ p$, then p is the sub formula; we must have the label $((\))$ so we are saying that if any state s is labeled with p , then label with it $AF\ p$.

Why you can do these things? Already we have defined the semantics and in our semantics; what we have seen? That fusel behavior includes the present scenario also so basically if you look in to time line say this is my present and this is my feature direction and this is my fast. Now while depending our semantics, what we have seen that, the feature behavior of the system includes the present behavior also, that means my feature is from this particular present scenario. So this is as per the semantics, that we have define since if p is true in presence state; we can say that little bit true in the feature also, so if any state s is labeled with p , we are going to labeled with $AF\ p$. So this is the first step we are doing, then we are going to repeat it like that label any state with $AF\ p$; if all successor states are labeled with $AF\ p$ until there is no change.

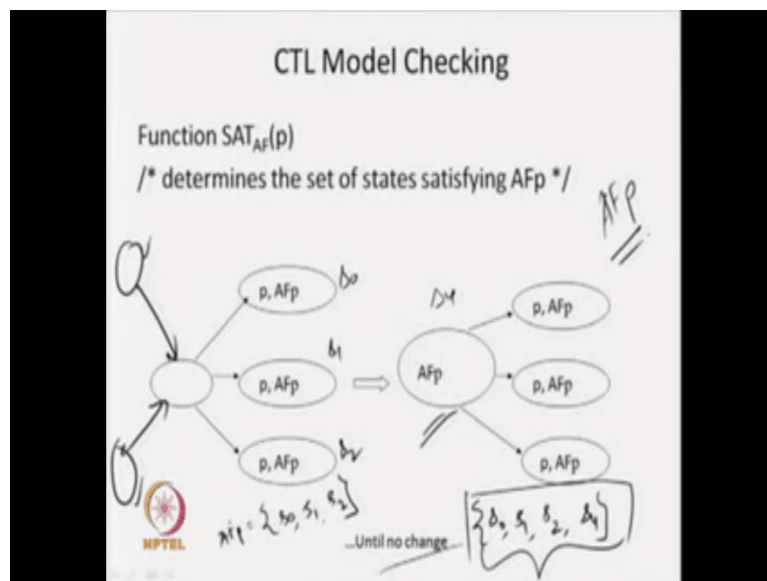
So know what will happen? First this is our best condition is that set of state where p is true, we are going to label those particular states with the formula $AF\ p$; then we are going to repeat our procedural label any state with $AF\ p$. If all successors' states are labeled with $AF\ p$ until there is no change, that means we are going to collect the states tortabilly and if you cannot include any more states, then we can stop at that particular point.

So this is showing that, until there is no change; so see why we can do these things? Because, we have seen one equivalence that $AF\ p$ we can write $AF\ p$ with the help of this equivalence; that p or $AX\ AF\ p$ what it says that, if p is true in a particular state, then will say that $AF\ p$ is true. So these particular p is captured by this particular my first condition; if any state s is labeled with p label it with $AF\ p$ and this is possible because my feature includes the present behavior and what the next component we are saying that, either p is true in this states or we are going to say that in all part next state $AF\ p$ is true. So that means if this is your either p is true ever here or whatever states we are going to get say here in all state, $AF\ p$ is true.

So in all part next state AF p is true so like that if it is that then we can include this particular set of states; where this given formula AF p is true. So that why, we are saying that label any state with AF p, if all successor states are labeled with AF p; we are iteratively going to labeled with so if one labeling those particular states with your AF p because if p is true in all those particular state, then we can very well labeled with your AF p.

So in this particular case we can label this particular state with AF p so that is why we can say that and like that we are going to collect each and every states; that is why you are saying that, we can repeat this particular procedure until there is no change, no change of the set of states that we have collected. So this is done with respect to this particular equivalence; if p is true that means, this is the first step and we are repeating this particular second step for that particular portion.

(Refer Slide Time: 32:36)



Now you just see that this is the given level so we are saying that just as an example we are saying that these are the three states where p is true that means it is labeled with p and this one is not labeled with p. Now according to this particular first one, we are saying that if any state is labeled with p label with AF p. So since p is true in these particular three states so we are labeling with AF p see the fast step; then we are going to repeat our procedure.

So we are going to do it until there is no chance, than what will happen? Since we know that $AF p$ is true in this particular three states, we are going to look for this predecessors since this predecessor; we are getting on an on here we are saying that in all those particular three state $AF p$ is true. So we can label this particular state will also $AF p$; so this is the way that we are going to left, if we have some more states over here, then what will happen? So like that we can go backward say, if I can have some more state.

So once you know the label of this thing, then we can go for the label of those particular states also will proceed in this way in record manners and until there is no chance; that means we cannot collect any more state that means what happens? You just see that initially may this particular where is $AF p$ is true. I can say that, it is true is your s_0, s_1, s_2 say these are the three state s_0, s_1, s_2 . Now when we have down this particular state say this is my s_4 that means, the set of state where this particular formula is true becomes s_0, s_1, s_2 and say s_4 ; like that will proceed and they unless we cannot improve any more states in this particular state, then will stop there and we are going to written this particular set of states for saying that, these are the states where this particular formula $AF p$ is true.


(Refer Slide Time: 34:42)


CTL Model Checking

```

Function SATAF(p)
/* determines the set of states satisfying AFp */
local var X, Y
begin
  X := S, Y := SAT(p),
  repeat until X = Y
  begin
    X := Y
    Y := Y ∪ {s | for all s' with s → s' we have s' ∈ Y}
  end
  return Y

```





So this is the basic concept. Now after having this particular concept, now what we can do? We can look for the algorithm; so this is the simple algorithm, that we are having this set of $AF p$ is going to determines the set of states, where this particular formula AF

p is true. Now like your previous case, we are going to take help of two variable X and Y , and what is this X ? For initially we are initializing acts to all the steps, basically this is the complete as we are taking you are saying that, this is the complete states place you are initially saying true s and what is your Y ? Y is your calling this particular satisfying to function with this particular formula ϕ ; ϕ is a any CTL formula and it is the sub formula of this particular given formula. So basically we know the level of this particular formula p that means, when we call this particular procedure satisfiability with this particular sub formula p , it is going to returning these particular states were the given formula p is true.

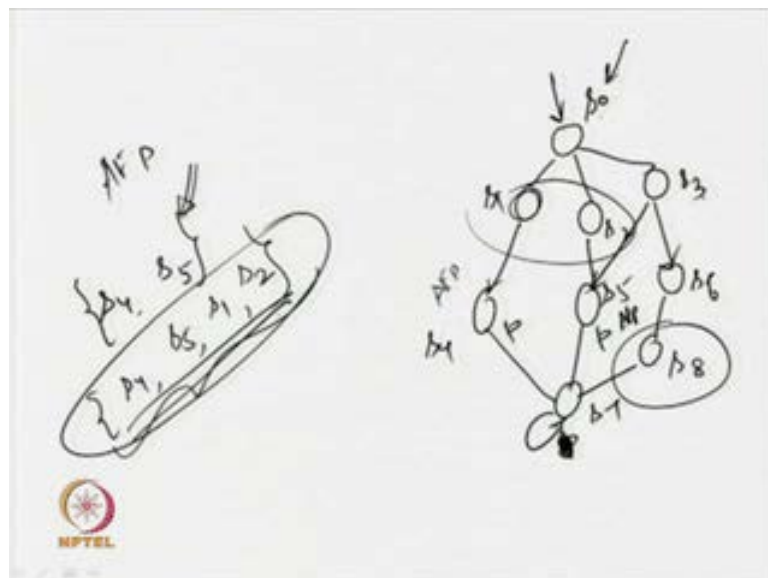
Now what happen? So whenever p is true we are say that, $AF\ p$ is true. So what we are doing? So we are saying that, now Y is assign to X ; so just saying that, initially in all those particular state Y the given formula AF is true. So this is in one state we have getting some state, where this particular formula AF is true so you have just keeping this particular information X . Now when we are going do this particular loop, until X equal to Y ; so initially we are having some state Y , where this particular formula p is true and this is the whole set states. So they may not be equal at the verification provided in all states are marked with p ; so until X equal to Y . We are going to repeat it so what we are doing? We are keeping the previous information because in all state Y AF is true, this is according to my first condition; if any state s is labeled with p and labeled it with $AF\ p$.

So this is my first condition; so according to first condition in all the states of this particular Y labeled with $AF\ p$ so we are getting this initial $(())$ and what we are going check? Now Y will be updated now; now what are the states that we can include in this particular state Y ? You just see that, if we are having some state of scenario; if all the next states all labeled with your $AF\ p$, then we can very well include this particular state to the set of states; where $AF\ p$ is true. So that is why, we are having this particular loop sometimes Y will be updated now; Y union, what are these things we are going to look for all such type of states, where we are having a transition from this particular s to all such type of s' and where s' is a member of Y .

So this is the basic similar actually, we are going to look for such type of state s' and we are going to look for all the transition and all the next states, that we are going to their $AF\ p$ must be true over there; so that means they are the member of this particular set Y . So that is why you are saying that, all such type of s' we are going to look and those

all s must be a member of Y ; that means, in all those particular s $AF p$ is true. So we are now appending this particular Y with those new states; again will repeat this particular loop and we are going to check whether X equal to Y or not. If in any of this particular scenario, if Y is not updated than X and Y remain same; so at that particular case we can terminal these particular procedure. You just see that, we are collecting going from the particular set of state, where p is true and traversing the whole graph and collecting all the set, if at any point of term we cannot include any more states; it is going to say that, now you stop here it shall because now there is no possibility of including any more states.

(Refer Slide Time: 38:53)



Because you just see that, I am value one example say we are looking for in alpha $AF p$ is true; do not say that, fuse with these two are marked with true. So these are the state $s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$. Now as per our this is algorithm, for the first step we are going to start with your s_4 and s_5 ; so these are the two states. So now this is the basic state, now we are starting with s_4 and s_5 . Now we are going to look for these particular predecessor state, now we are coming to this s_1 whenever you go in all so this will be labeled with your $AF p$; this will also labeled with $AF p$. So in all state I am coming to this particular state and we are saying that, we are getting one particular step s_4 ; only one state which is labeled with $AF p$. So I can include this one s_4, s_5 and I can include now this a s_1 also.

Similarly, when s_5 is also member over here, then what will happen? I am going to get one predecessor state. So I will give you $(())$ transition so we are giving coming to s_2 . So there is only one successor state, where it is labeled with your AF p ; so I can include this particular s_2 over here. Now this particular s_5 is using another predecessor s_3 . Now when we coming to s_3 will find that, it is having two successors; one is labeled with AF p , one is not labeled with AF p . So s_3 cannot be with that so we are coming to this particular side. Now in this particular cause what will happen? Now Y is updated from s_4, s_5 to s_4, s_5, s_1, s_2 .

Now will go in to the loop; now what will happen in this particular cause? Now these two numbers are you need that, we are going to look for the predecessor; so s_1, s_0 . Now in s_0 , we are having three successor s_1, s_2 and s_3 ; s_1 and s_2 are labeled with your AF p , but s_3 is not labeled with your AF p so we cannot label s_0 with AF p . So in this particular case my states remaining same; so there is no sense so at that particular point we can stop of procedure and we have said that, these are the course that we are going to include because in this particular case since no more states are including over here.

So there is no possibility of getting any more state because we are going to see all the predecessors; that means, if someone is not member over here that means, it is not a member of your $(())$ like that say s_8 . If you said this thing so what will happen? From s_8 we will go to s_7 and p is not true over here. So we are not going to get any part where in future p is true; otherwise, if p is true here than in the very first case p that include over here also. So there is no sense that anymore states will be included over here; so if there is no sense in this particular set of states, that we can stop that procedure here itself. So that is why you are saying that, until there is no sense. So eventually we are coming up with a very elegant procedure so this procedure can be now implemented; first collect all the set of state where p is true, then repeat this particular loops.

Now another operator, that temporal operator or CTL operator that we need is your E p until q . Now how we are going to say that, E p until q they are exist a path where p remains true until q becomes true. Now again you see that according our semantics, that future includes the present behavior. So that is why the first step is coming, if any state s is labeled with q , labeled it with E p until q because future includes the present behavior. So if any states is labeled with your q we can say that, if p until q is true over here; so we are going to label this particular states with E p until q .

(Refer Slide Time: 42:46)

CTL Model Checking

Temporal Operator:
 $E(p \ U \ q)$

- If any state s is labeled with q , label it with $E(p \ U \ q)$
- Repeat: label any state with $E(p \ U \ q)$ if it is labeled with p and at least one of its successor is labeled with $E(p \ U \ q)$ until there is no change.

$E[p \ U \ q] = q \vee (p \wedge EX \ E[p \ U \ q])$

NPTEL

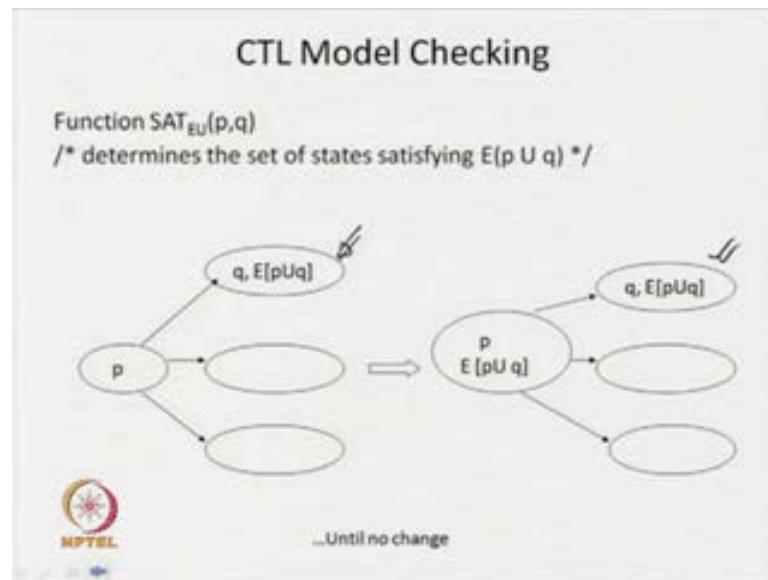
Now we are going to repeat this particular procedure label any state with $E \ p \ \text{until} \ q$; if it is labeled with p and at least one of its successor is labeled with $E \ p \ \text{until} \ q$. So p should remains to until q becomes true and we need at least one part; so that is why I saying that at least one of its successor is labeled with $E \ p \ \text{until} \ q$ and p must be true at that particular part. So if the state is labeled with p and if we find that any of the successors is labeled with $E \ p \ \text{until} \ q$; then we can label this particular state with $E \ p \ \text{until} \ q$ because if any one of the successor is labeled with $E \ p \ \text{until} \ q$, at least we will say that in future somewhere q is true and till that particular point p remains true.

So that is why we can say that, past is any state s is labeled with q ; we are going to labeled with $E \ p \ \text{until} \ q$ and we are going to repeat this procedure. Why this possible? Because, we seen this particular equivalence; all ready we have seen this particular equivalent. What we are saying that, $E \ p \ \text{until} \ q$ can be expresses q or p and they are exist a part next state $E \ p \ \text{until} \ q$. So this is you just see that, if p is q is true in as we are going to labeled with $p \ \text{until} \ q$. So this is the first part first decision, if q is true there then we are going to labeled with $E \ p \ \text{until} \ q$.

So if q is true then fine, we are going to get those particular states and if it is said or we are going to have this particular condition. So this is the second clause or that repeat one; p must be true in that particular state and there exist a path in next state, it is labeled with $p \ \text{until} \ q$. So from this particular equivalent we are developing these particular methods

and we are going to repeat it until there is no sense. So we are going to collect states one after another and we are going to stop in a particular point; when we cannot include any more states. So we are going to say that until there is no sense so this is the equivalence; from this particular equivalent we have got an method for E p until q.

(Refer Slide Time: 45:51)



So you just see that simple example; I am saying that similar example I am taking, say these are the three states; initially we are having say s_0, s_1, s_2, s_3 . Now as for the first condition, wherever q is true we are going to be labeled with E p until q. So by looking into these things what will happen? We can label this particular state s_1 with p E p until q. So this is the only state where q is true; so we can be labeled with it p until q, then what will happen?


Once we know the label we can traverse the graph and we can say that, we are going to look for this particular states; we are going to look for the predecessor here and if this is labeled with p then we can be labeled it with p until q because it is labeled with p and any one of this particular successors is labeled with E p until q; say p remains to until q becomes true. So since q is true we are labeling with your p until q and we have come to this particular state and we have seen that p is true so we can be labeled with it E p until q. So this is the way that we are going to traverse the graph until there is no sense.

(Refer Slide Time: 47:04)

CTL Model Checking

```
Function SATCTL(p,q)
/* determines the set of states satisfying E(p U q) */
local var W,X,Y
begin
  W := SAT(p), X := S, Y := SAT(q)
  repeat until X = Y
  begin
    X := Y
    Y := Y ∪ (W ∩ {s | exists s' such that s → s' and s' ∈ Y})
  end
  return Y
end
```

p, q
 $E(p U q)$
 $Y: ?$
 $W: ?$



Now look into the procedures so we are going a Nelligan procedure for this particular E p until q also; so we are going to have a function satisfiability E until p and q. So we are going to take now three variables W, X, Y. Now we are having sub formula p and q because we are going look for E p until q; so this is one sub formula; this is another sub formula. So in W we are going to say that, going to call this particular satisfiability function with p because we know we must know the level of p. So this is the said W; X is basically we are going to send the entire states place space X is equal to s and Y is your satisfiability of this particular q, sub formula q.

So we are going to collect all the states where q is true and we are going to say that, this is my particular set Y. So Y is in set of state where q is true and W is your set of state where this particular formula p is true. Now we are going repeat this particular loop; now what it says that? Y if you are going to look into Y is the states where this q is true and we have seen that, where ever q is true we are going say that E p until q is also closure. If any state s is labeled with q, than we are going labeled with E p until q and this is because of our definition of semantics because future in step of present behavior.

So that is why, this is the set of state where E p until q is true. Now we are going to collect the states where it is true and we are going to apply this particular state Y. Now what will happen? Since this is a set of state, now we are going to say that these are the previous condition we are going to keep; so in set of state Y p until q, E p until q is true.

So we are just preserving this particular set of state within X; now as I tend to bring s 2 those particular states.

Now we are going to open this particular Y; so how we are opening it? This is your Y union that means; now we are going to look for those particular states, where our given or required condition is satisfied. So it should be sustainable like that W; what is W? This is the set of state where p is true. So W intersection we are going to collect all such type of state s such that, from s to s thus we are having a transition and this particular s dash is a member of Y; that means, in s dash we are going to see that, it is already E p until q is already true in this particular state of s dash. So if we are having a transition from this particular state s to the set of states, where E p until q is true.

So any states where E p until q is true and we are having a transition from s and we can say that in s E p until q is true. So that is why we are appending it; so we are and second condition is in s, p must be true; so that is why whatever states we are collecting, along with that you are intersecting with the W; in W it says that the set of state where p is true so that step p must be true and whatever states we are getting, whatever we are appending with this particular point and where going back to this particular repeat step; one we are going to check whether X is equal to Y or not. What we have in X? These is the previous set of state, where your E p until q is true; now so we are going to check whether any new more your states has been added or not; if it is added that means, these are not equal than again will go in to this particular loop and we are going to collect.

So, you just see that, this is the procedure we are having and this is nothing but the graph traversal algorithm. For you are going to start from some states of this particular graph and we are going to look for all the predecessor, if my given conditions is true or given situation is true in this particular predecessor. We are going to include those particular state also in my set of states. Then again I will going to repeat these things, so what about new states we are including? We are going to form those particular state, again we are going to look for all the predecessor. And if any new more states cannot be added to these particular state, then we can say that we are not going to any more state and this is the final set of state we are going to get; and that is why we are returning this particular set of state Y. So these are all set of state we are going to say.

(Refer Slide Time: 51:55)

The slide is titled "CTL Model Checking". It contains a bullet point: "• After performing the labeling for all the subformulas of Φ (including Φ itself), we output the states which are labeled Φ .". In the bottom right corner, there is a hand-drawn circle containing the text "EX", "AF", and "AU" stacked vertically. In the bottom left corner, there is a logo for NPTEL.



So in our CTL model checking algorithm now, what we are going to have? We are saying that after performing the labeling for all the sub formulas ϕ , including ϕ itself we output the set of state which are labeled with ϕ . So finally, we are going to give those particular set of state while this true. So you just see that you have to, you need the minimum set of operators; so we are considering EX, AF and AU. So we are simply seen how these can be implemented then we have got very Elgin algorithms Elgin procedures to implement these three operators. So once we need know the algorithm for these three operators. Now address can be express with the help of these three. Now what will happen now we have to work for, look for each and every sub formula; and once you know the sub formula, then only we can go for a main formula.

So that is why we can say that, what is the time require to do this particular model checking. If you find that we can say that. Time complexity of the algorithm will be $O(f \text{ and set of state and this is the complete graph } V \text{ plus } E)$; that means, set of state and set of transition that we have. This is basically I have say that, number of connectives in the formula; that means number of sub formula that we have. Basically you just see that, if am going to say that ϕ_1 and ϕ_2 . I have to call this algorithm in two times; because first I have look for the ϕ_1 , then we have to look for a ϕ_2 . We know the labeling of these two, then only I go to the consumption these two things.

(Refer Slide Time: 52:57)

CTL Model Checking

- After performing the labeling for all the subformulas of Φ (including Φ itself), we output the states which are labeled Φ .
- The complexity of the algorithm is
 - $O(|f| \cdot |V| \cdot (V+E))$
 - f : number of connectives in the formula
 - V : number of states
 - E : number of transitions



So that is why, it is the number of connectives or we can say length of the formula that we have over here. Because when I give this particular formula, we must know truth values of ϕ_1 and ϕ_2 ; then only we can look this one. And you see that, first you have collecting some particular states, from that particular state, we need to see all the predecessor state. In EX, this is only one label, but in case of AF and AU we have to iteratively we have look for this particular things.

So from all states at least we have to look for this predecessor. And E is the number of transition; that means words scenerition, words scenario we have to refer the complete graph traversal. That is why we are saying that this is the complexity of my labeling algorithm. So thing this your linear to the length of the and this say that quadratic to the number of knots, that we have in this particular graphs. So this is simple graph traversal algorithm and the time complexity depends on the length of my formula because for every sub formula I have to call this particular procedures. And we have to look for each and every state of this particular inverse in add we have to look for each and every state of this particular graph. So this is basically we are going to have an linier time algorithm for this particular procedures.

So this is the beauty of your model, CTL model checking algorithm, so it easy to automate. These are the basic three algorithm, we have seen and next class we are going to see some example will explain; and you see how we are going to or we will see the

execution trace of this particular algorithm. In next class again we will going to discuss about these three operators only with the help of one examples. I will stop here today.