Embedded Systems - Design Verification and Test

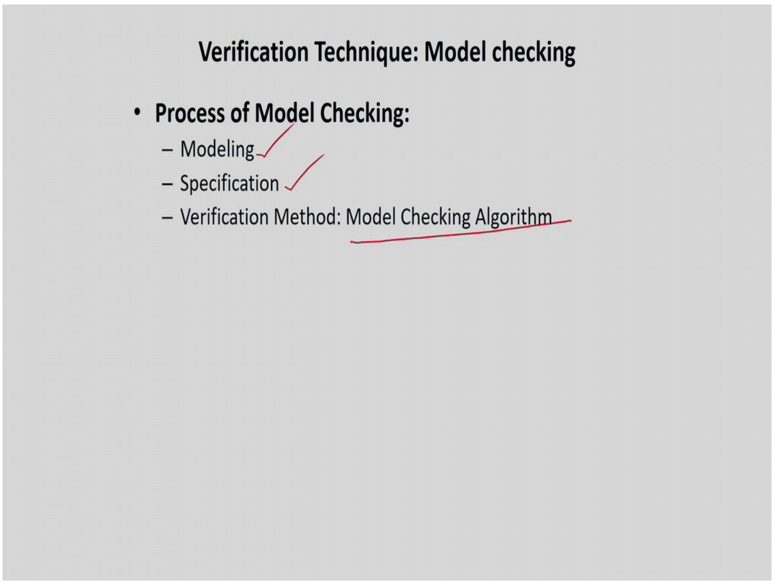
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture - 22
Model Checking Algorithm

Welcome back to the online course on Embedded System Design Verification and Test. We have discussed about the temporal logic and in particular about CTL, Computational Tree Logic and we have seen how some of the property can be expressed with the help of your computational tree logic.

Now, we are going to look for a verification method which is known as model checking. So, model checking is a method by which we can check the correctness of a properties specified in CTL in a model.

(Refer Slide Time: 01:10)



Verification Technique: Model checking

- Process of Model Checking:
  - Modeling
  - Specification
  - Verification Method: Model Checking Algorithm

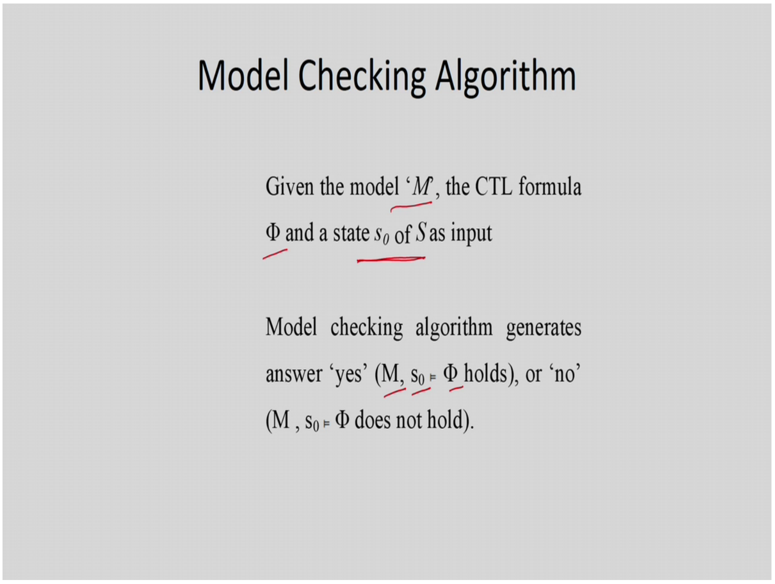
So, what is the verification techniques? Basically we are going to verify the specification of a given system in the design or the abstract design that we have come up.

So, today we are going to discuss about the model checking which is a verification technique. So what are the process of model checking? So, basically already we have mentioned that in verification technique basically we have one component which is known as the system model. We are designing our system and we have to represent our system with the help of some formal mechanism, and generally we use some finite state machine to represent our system. So, we need one model, our system and the model will be represented with the help of Kripke structure. Kripke structure is nothing but some sort of a finite state machine that already we have seen and the specification of the system will be given or represent with the help of some property specification logic. And in our case we have discussed about CTL Computational Tree Logic.

Now, first we are going to have the model of our system then we are going to write our specification of property in some formal way. And finally, we are going to have a verification method, and in our verification method we are going to talk about model checking algorithm.

Today we are going to discuss about this model checking algorithm, how we are going to verify the property specified our return in CTL. I am going to check the truth of this property in a model that we have designed for our system. So now, how we can visualize the model checking algorithm?

(Refer Slide Time: 03:05)



Model Checking Algorithm

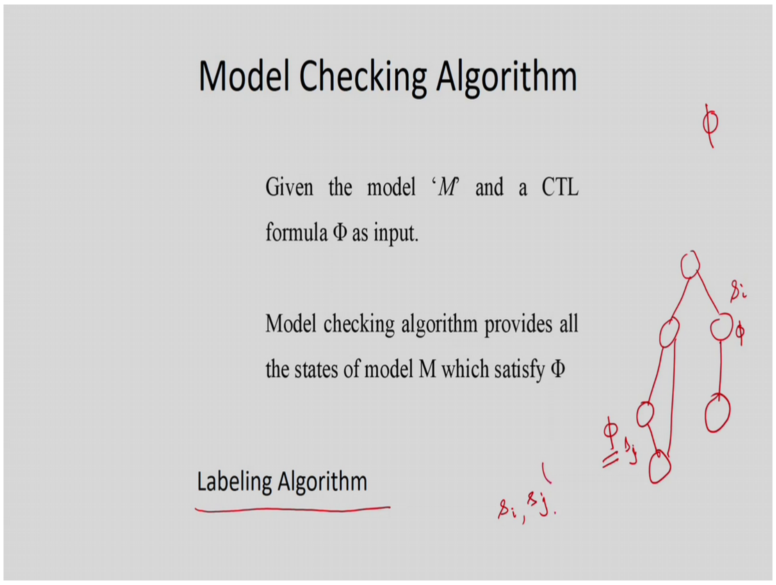
Given the model '*M*', the CTL formula Φ and a state s_0 of *S* as input

Model checking algorithm generates answer 'yes' (M, $s_0 \models \Phi$ holds), or 'no' (M , $s_0 \models \Phi$ does not hold).

So, one way we can say that given a given a model M and the CTL phi and a state s 0 of S. So, here we are going to have 3 input we are going to talk about a model M, we are going to talk about one CTL formula and a particular state s 0 or maybe a s i of the system. So, these 3 will be input to our model checking algorithm.

So, what model checking algorithm says? It is going to give me yes if the property phi is true in an in the state s 0 of model M or otherwise it will say no if this property does not hold in that particular state s 0 of the model M. This is one way we can represent it. What are the inputs? It is having 3 inputs; model M, state of a model and property. And what model checking algorithms or method gives us? It says yes if the property is true in the state s 0 or it says no if the property is not true in the state s 0. So, these are the outcome of our model checking algorithm.

(Refer Slide Time: 04:19)



Model Checking Algorithm

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

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

Labeling Algorithm

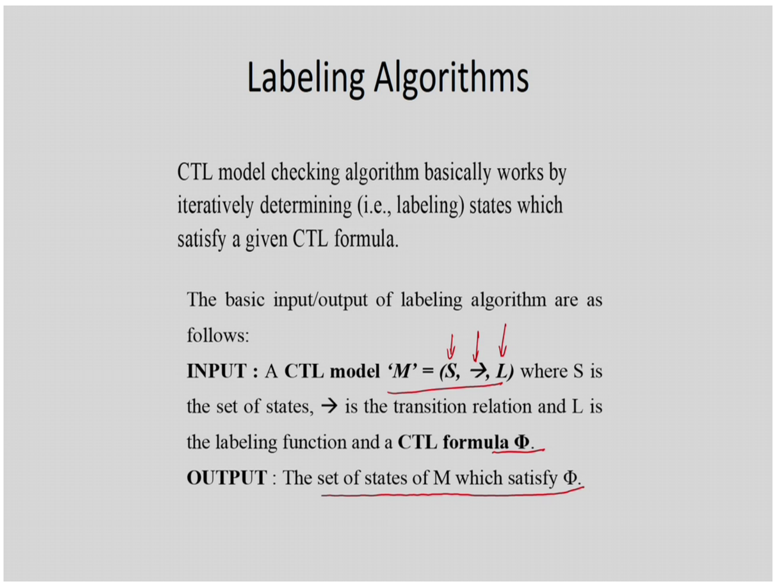
ϕ
s_i
ϕ
ϕ
s_j
s_i, s_j

In that model checking algorithm can be again treated in a different way. So, in this case what we are going to say we are going to give a model M and a CTL formula phi. So, we are having a method model checking method. What are inputs? Model M and the specification returning our CTL formula phi now, what model checking algorithm gives? It gives us all the states of model M which satisfy phi. So, in the model M we are going to check for the property phi and as a result this model checking algorithm give us the all the states where this particular formula is true.

So, we are going to discuss this particular approach that you are going to give a model M and the formula phi and this method model checking method returns as all the state where the given formula phi is true. So, basically this approach is nothing but some sort of labeling algorithm. We can say we are going to label each and every state with the help of the given CTL formula if the CTL formula is true at that particular state. So, basically this model checking algorithm is nothing but a kind of your labeling algorithm we are going to label it.

So, if say, if this is the model some sort of model that we have, and say phi is a formula, if phi is true in this particular state then we are going to label this state with this particular given formula phi. Now, we will see how we are going to label this particular phi or how we are going to check that truth values of this particular phi. If the truth values of the phi is true in this particular state say s i then it will be labeled phi on the other hand. Now, these as result this labeling algorithm is going to return this particular state s i, s j is the set of state where a given formula phi is true.

(Refer Slide Time: 06:27)



Labeling Algorithms

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

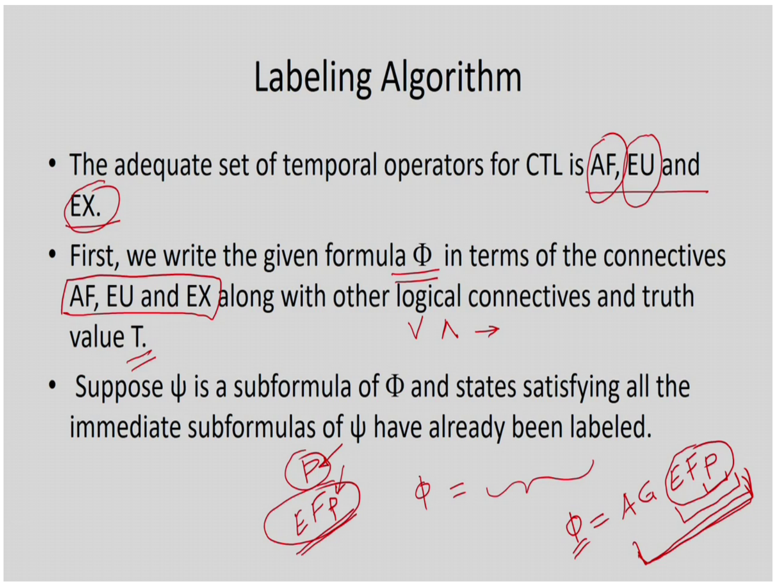
The basic input/output of labeling algorithm are as follows:

INPUT : A **CTL model** ***'M'* = *(S, →, L)*** where S is the set of states, → 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, now we are saying that CTL model checking algorithm basically works by iteratively determining or labeling state which satisfies the given formula phi. So, iteratively it will going to check each and every state and if the formula is true then it will done to label with that particular formula and after labeling all the stats it is going to return as a set of states where we have the label of phi that means, where the state where the formula phi is true. So, the input is the given model M and the CTL formula phi and it returns what is the output set of state of M where which satisfy phi. So, two inputs we are giving and as a result as an output we are getting a set of state.

(Refer Slide Time: 07:15)



Labeling Algorithm

- 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.

$\vee \ \wedge \ \rightarrow$
P
EFP
$\phi =$
$\phi = AG\ EFP$

Now, what we are going to look for? Labeling algorithm, already we have discussed that in CTL we are having temporal operators and temporal operators are preceded by part quantifier A and E, and we are having several CTL operators but already we have seen that there is an adequate set of operators with the help of this adequate set of operators other operators can be derived.

So, in this particular case we are going to look for the adequate set of operators says your AF in all path in future phi EU, there exist a path with until operator and EX there exist a path in next state. So, these are the 3 basic operator we are going to look into it and we are going to find out a method to check for the truth of this particular formula in a given model.

Now, in this particular case what (Refer Time: 08:12) if we give any formula phi then we are going to convert this particular formula or we are going to get an equivalent formula for phi which involves only these 3 temporal operator AF, EX and EU. Because this is the your adequate set of temporal operator we are considering. And along with that logical connectives because, you may have that conjunction disjunction conjunction implication all those logical connectives we have and along with the truth value true or top.

So, if suppose psi is a sub formula of phi and state satisfying all the immediate sub formulas of psi have already been labeled. Now, we are going by label by label or we can say that if it is a given formula we are going from all the sub formulas. First we have to look for the primitive one and how we are going to get the primitive one we discuss about it. And we know about the labeling function of our model because the model consist of a labeling function also because this is a set of state, this is the transition relation and L is a labeling function with the help of labeling function we know truth values of atomic proposition in the states.

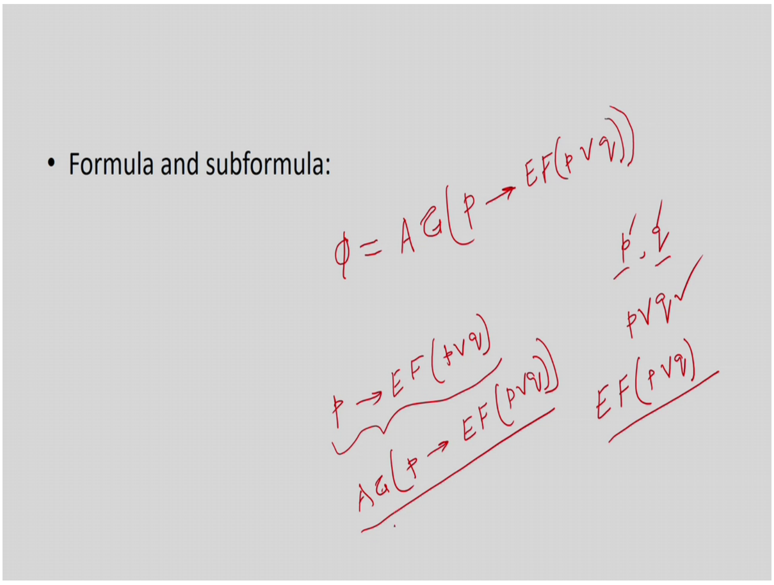
So, now if phi is a formula then phi can compose of several sub formulas. Now, we have to look for each and every sub formula and if we know the truth values of a sub formula then only we can look for the truth values of the given formula. So, as for example, I can say that if I say that AG there exist a path in future p.

So, in this particular formula what we are talking about? The notion of sub formula p is a atomic proposition. So, p itself is a sub formula and p is an atomic formula. So, it is a CTL formula so, EF p is another CTL formula because if I know that if phi is a CTL formula then EF phi is also CTL formula.

Now, AG EF p is the CTL formula because if phi is a CTL formula then AG phi is also CTL formula. So, in this particular case if phi this is the given formula then what are the sub formula we are getting over here, p is a CTL formula which is the sub formula of this given CTL formula. Then second one is your EF p which is another sub formula and this is the given another formula. So, to find out the truth values of phi first of all we have to know the truth values of the sub formula EF p also to know the truth values of this particular sub formula EF p we must know that truth values of this particular p the atomic proposition. So, this is atomic proposition and atomic proposition is going to take truth values either true and false.

So, in this particular case you just see how we are going to get the truth values of p? Through the labelling function, because in our model we are having a labeling function and we are going to label the states where the atomic proposition is true. So, we know the set of states where the truth value of p is true. Now, with the help of this things. Now, we are going to find out EF p because we know the truth values of p. So, we can find out the set of state where the truth values is true for this particular formula EF p. When we know the truth values of this particular EF p that means, if we know the set of state where this particular sub formula EF p is true, then we can look for the set of state where the given formula AG EF phi is true. So, we are going label by label or say sub formula by sub formula to look for the truth values of a given CTL formula.

(Refer Slide Time: 12:22)



- Formula and subformula:

$\phi = AG(p \to EF(p \vee q))$

p, q

$p \vee q$ ✓

$EF(p \vee q)$

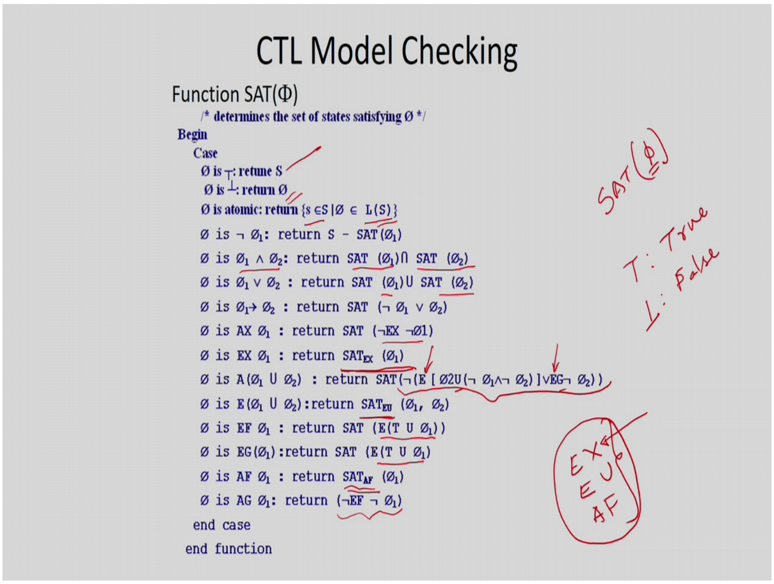
$\underbrace{p \to EF(p \vee q)}$

$AG(p \to EF(p \vee q))$

So, already I have mentioned about this formula and sub formulas because I have already mentioned about say if I can say like that another sub formula if I take in all path globally p implies there exist a path in future p or q. So, in this particular case you just see we are having a atomic proposition p and q. So, these are the sub formulas, we know the set of states where these two sub formulas are true or the atomic proposition p and q are true we know these things with the help of labelling function since p and q are your atomic proposition. So, p or q is say atomic proposition, if we know truth values of p and q then we can find out the truth values of p and q and we will be knowing the set of states where this particular p and q is true.

Now, if you know the truth values of p and q are the set of state where p and q is true, p or q is true then we can look for the truth values of EF p or q. So, once you know the set of state where EF p or q is true then we can look for the set of state where the sub formula EF p or q. So, once we know the set of state where this particular sub formula is true then we can look for the set of states, where AG p implies there exist a path in future p of q is true. So, these are the atomic proposition then we are going to look for the sub formula p or q then EF p or q and p implies EF p or q and finally, AG p implies EF p or q.

(Refer Slide Time: 14:28)



CTL Model Checking

```
Function SAT(Φ)
    /* determines the set of states satisfying Ø */
Begin
  Case
    Ø is ⊤: retune S
    Ø is ⊥: return Ø
    Ø is atomic: return {s ∈S |Ø ∈ L(S)}
    Ø is ¬ Ø1: return S - SAT(Ø1)
    Ø is Ø1 ∧ Ø2: return SAT (Ø1)∩ SAT (Ø2)
    Ø is Ø1 ∨ Ø2 : return SAT (Ø1)∪ SAT (Ø2)
    Ø is Ø1→ Ø2 : return SAT (¬ Ø1 ∨ Ø2)
    Ø is AX Ø1 : return SAT (¬EX ¬Ø1)
    Ø is EX Ø1 : return SATEX (Ø1)
    Ø is A(Ø1 U Ø2) : return SAT(¬(E [ Ø2U(¬ Ø1∧¬ Ø2) ]∨EG¬ Ø2))
    Ø is E(Ø1 U Ø2):return SATEU (Ø1, Ø2)
    Ø is EF Ø1 : return SAT (E(T U Ø1))
    Ø is EG(Ø1):return SAT (E(T U Ø1)
    Ø is AF Ø1 : return SATAF (Ø1)
    Ø is AG Ø1: return (¬EF ¬ Ø1)
  end case
 end function
```

So, now what is the model checking algorithm, CTL model checking algorithm? So, we are going to have a function call set of phi. So, this is basically satisfiable function and what is the input a CTL formula. So, we are going to now, find of the set of states where this particular given formula is true. And we are going to look for a some operators only which is adequate to express all other CTL operators. So, that is why you are saying that in that particular case we say that if phi is top already I said that top is nothing but the truth value true and bottom is nothing but the truth value false.

So, it says that if phi is set of truth value true which is a constant then it returns this particular all states, because in all state truth value true is true. If the given atomic for given formula phi is your bottom that means, truth value false then it will returns the null set because in none of the state false is true. So, if phi is an atomic proposition then what it returns? It returns all the states where phi is a member of the labelling function. We know then if it is your phi is naught of phi 1 then it returns s minus satisfiability of phi 1 because naught of phi is negation out of phi if in some states phi is true then the in remaining state naught of phi is true. So, it is going to be done s minus satisfiability of phi 1.

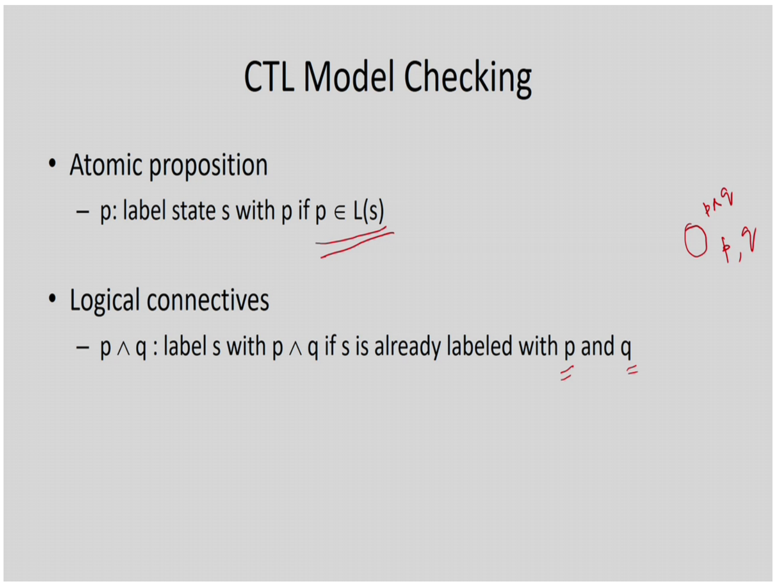
Similarly, it is a simple and connective, so satisfiability phi the set of state where phi 1 is true intersection set of state where phi 2 is true or so this is your union set of state where phi 1 is true or union set of state where phi 2 is true. So, phi 1 implies phi 2 then this is basically naught of phi 1 or phi 2 this is the equivalence. So, we are going to find out the set of state where naught of phi 1 or phi 2 is true AX phi 1 and we know that AX phi 1 can be expressed with naught of EX naught of phi 1. So, in that particular case we have to first write this expression with its equivalence expression. Now, we have to find out the truth values or going to look for the set of state where this particular formula is true.

So, here we are having one sub formula naught of phi 1 then EX phi 1 then naught of X phi 1. So, if phi EX phi 1 then it is going to return a satisfiability EX phi 1. So, this is a method we must have and we are going to discuss about this method. Similarly if phi is a phi 1 until phi 2 then we are going to look for the equivalence of this a phi 1 until phi 2 which is expressed with the help of there exist with until and globally.

So, first A phi 1 until phi 2 will be represent with the help of sub formula we are going to look for a truth values of this particular formula E phi 1 until phi 2. So, we have to get the method to find out the set of state where E phi 1 until phi 2 is true. Similarly EF phi 1 it will be represented its equivalent formula, EG phi 1 it will be represented with its equivalent formula. If AF phi 1 then we must have a method for AF where it is going to return a set of state where phi 1 is true and for AG phi 1 again we are going to look for the equivalent of this particular formula.

So, basically you just see we are talking about EX, EU and AF. These are the few operators which is the adequate set of operators. So, we are going to discuss about the methods to check for the truth values of these 3 operators in a given model and by discuss what we are going to say that we are going to label the set of states or label the states where this particular formula is true and finally, we can return those particular states by talking about that these formulas are true on this particular state. So, we are going to discuss about these 3 methods.

(Refer Slide Time: 19:16)

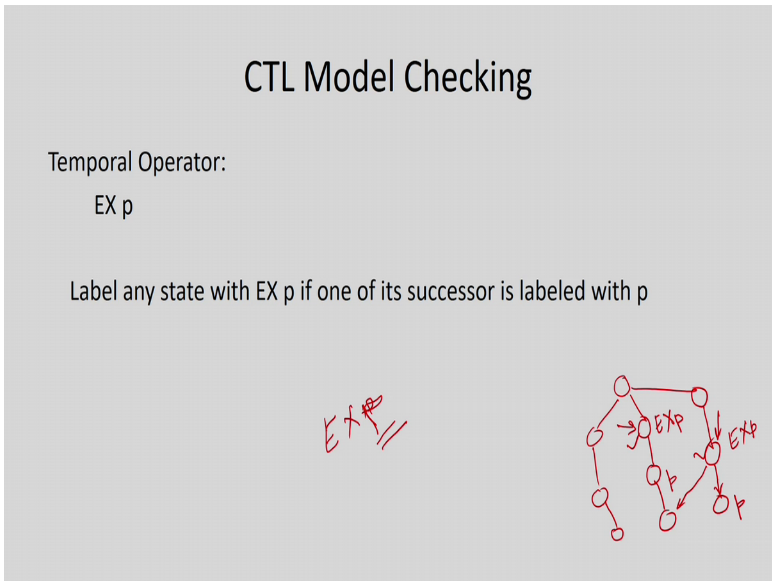


CTL Model Checking

- Atomic proposition
  - p: label state s with p if p ∈ L(s)
- Logical connectives
  - p ∧ q : label s with p ∧ q if s is already labeled with p and q

So, already we have discussed how we are going to talk about the atomic proposition it will be true in a set of states, in a state s if p is a member of this particular labelling points. Similarly logical connectives already we have said that if we know the state s if state is labelled with p and q, then already we have to set those states are labelled with p and q. So, if these state is labelled with p and q that means, if both p and q is true over here then we can labelled it with p and q this is the notion basically. So, like that other connectives also we can see.

(Refer Slide Time: 19:58)



CTL Model Checking

Temporal Operator:

EX p

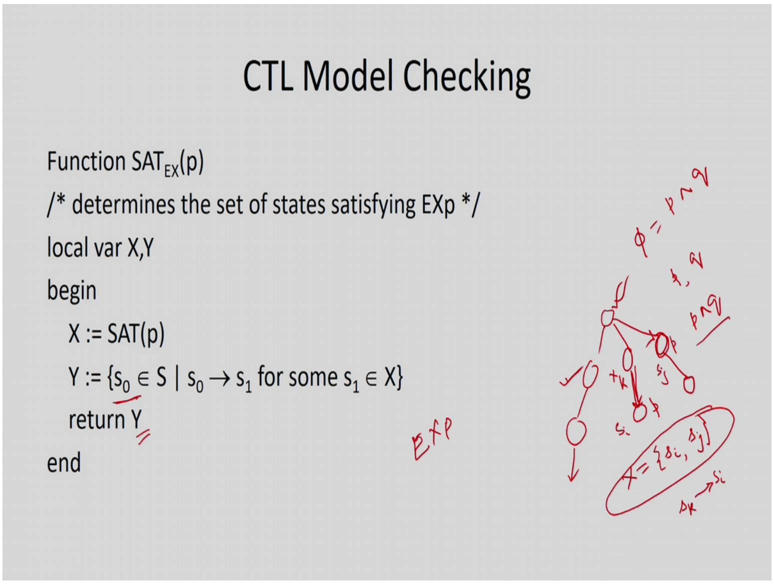
Label any state with EX p if one of its successor is labeled with p

EXp
EXp
EXp
p
p

Now, we are going to basically discuss about the temporal operators. So, first one is your EX p, there exist a path in next step p is true. So, label any state with EX p if one of its accessories labelled with p. Now, say first of all if we are going to consider about any system. And if we are going to look for this particular formula EX p then what will happen? First of all we must know the label of this particular atomic proposition p just see that if I am going to say that these are the two states where p is true and in all other states p is false.

So, according to the definition of EX p we know that EX p will be true in this particular state because there exist a next state, where p is true and similarly EX p will be true over here so that means, after looking for truth values of EX p it is going to label these two states with the formula EX p. Or on the other we can say the at the end of this particular algorithm it is going to return these two states and says that in these two states EX p is true. Now, how we are going to implement it or going to get a method for EX p?

(Refer Slide Time: 21:34)



CTL Model Checking

```
Function SAT_EX(p)
/* determines the set of states satisfying EXp */
local var X,Y
begin
   X := SAT(p)
   Y := {s_0 ∈ S | s_0 → s_1 for some s_1 ∈ X}
   return Y
end
```

$\phi = p \wedge q$
p, q
$p \wedge q$
EXp
$X = \{s_i, s_j\}$
$s_k \to s_i$

So, the function EX p we are going to define now. So, it is going to look for the set of state where p is satisfying, where p is true p maybe now in any CTL formula, it may not be atomic proposition it may be any CTL formula. Because already I have said that if my CTL formula phi is your p and q, then first of all we must know about the truth values of p and q and once we know the truth values of p and q then we know the truth values of p and q or we know the set of state where p and q is true. So, this p is any CTL formula.

Now, determine the set of state that satisfy EX p. Now, what we are going to say? We are going to take two set X and Y, these are the variable local variable X and Y. Now, what we are going to say? Plus X is nothing but the set of state, where p is true. So, we know that if in a given model already in the last slide also I have mentioned, if it is mark say s i and s j is marked with p or labelled with p then satisfiability of p is going to give me the set of state s i, s j.

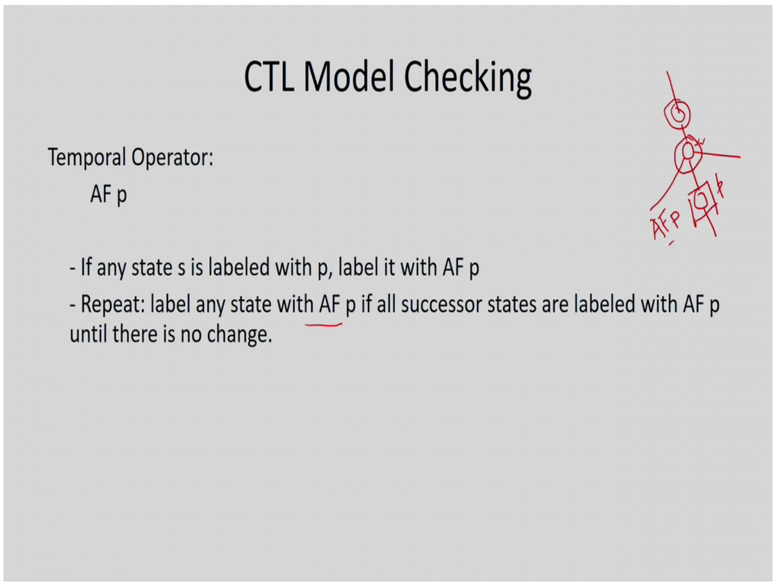
Then we are going to now connect the states, why? What we are going to why we are going to connect the state s 0 belongs to s. So, belongs to this particular state space. Now, what are the properties of this s 0? It says that we are having a transition from s 0 to s 1 we are going to have a transition from s 0 to s 1 for some s 1 belongs to X. Now, in this particular case this is X, set of state where p is true. So, s i and s j is going to have X.

Now, we are going to look for such type of states s 0 where I am having a transition from s 0 to any of the state of this particular s i or s j. So, if this is X k, then what will happen? At least we are going to get one transition next state is s j. So, I am having this s k to s i such that in s i p is true. So, s k will come returning similarly this particular state will also come into picture because from here at least I am going to get one state, where p is true but if I am going to look into this particular state then it will not come into picture because in the next step p is not true. So, it is going to return those particular states and final we are going to return Y.

So, what it basically does? It is going to return with a set of states where EX p is true we are looking for the formula EX p. So, this is the way that we can say this is the very simple one, how we are doing? By looking into the model, model is nothing but some sort of final state machine which is basically similar to a graph. So, it is a nothing but a graph proposition algorithm.

First we are connecting those particular set of states where the given formula is true then we are just looking for its predecessors and we are connecting all the predecessor and we are returning those particular predecessor. So, these are the set of state where EX p is true. So, already I have explained with example.

(Refer Slide Time: 25:20)



CTL Model Checking

Temporal Operator:

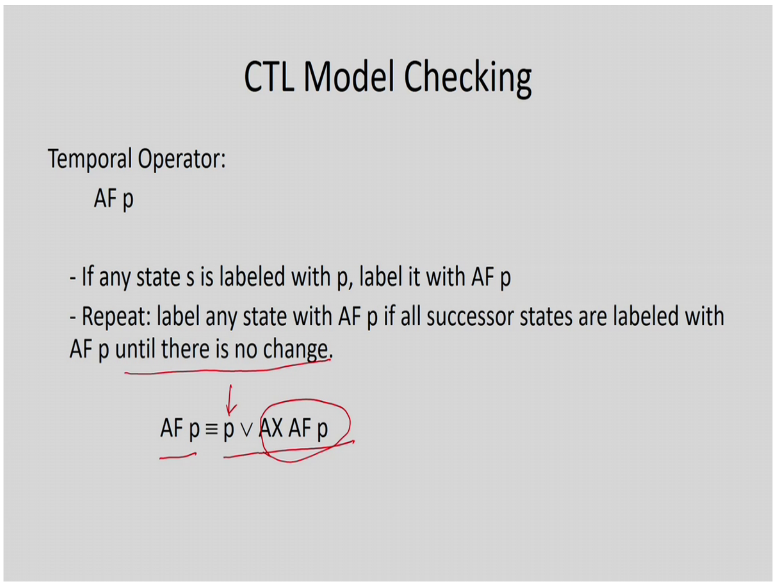
AF p

- 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.

Now, second temporal operator I am going to say AF p in all path in future p holds. So, in that particular case we just say how we are going to say if any state s is labelled with p then label it with AF p. So, if any state p is true then I am going to labelled it with AF p. Why you are doing it? Because as per our different semantics in our different semantics what we are saying that future includes present, since future includes present behaviour. So, that is why I can say that AF p is true over here.

Then what we are doing? Wherever p is true we are labelling with AF p then we are saying repeat label any state with AF p if all successor states are labelled with AF p. So, if it is labelled with AF p, then I can label this one also AF p provided all its successors are labeled with AF p until there is no change, until there is no change. Now, thus it means first starting from this one then I am connecting this state then I am going to look for this predecessor and I am going to look for all the transition of the given graph and if we cannot add anymore state then we stopped here. So, repeat until any sense means until there is no change that means, we are not going to get any new more states.

(Refer Slide Time: 27:01)



CTL Model Checking

Temporal Operator:

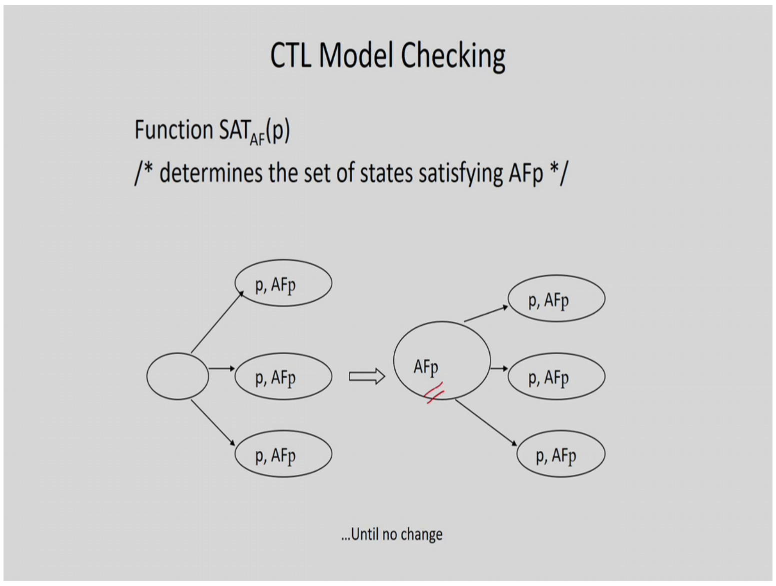
AF p

- 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.

$$AF\ p \equiv p \lor AX\ AF\ p$$

So, how we are doing it? So, how it is possible? So, basically already we have discussed about the equivalence AF p is equivalent to AF p or AX AF p. If p is true in the state then we can say that AF p is true or if p is not true in a particular state then I have to see AX AF p must be true in that particular state. So, every iteration we are going to connect more and more states and finally, we cannot connect any more state then we are going to stop our algorithm at that particular path. So, that is why it says until there is no sense.

(Refer Slide Time: 27:40)



CTL Model Checking

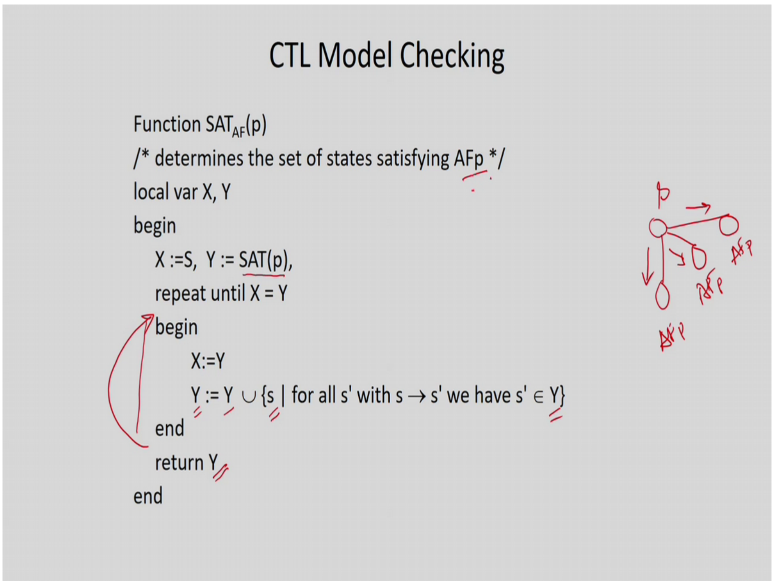
Function $SAT_{AF}(p)$

/* determines the set of states satisfying AFp */

So, now with the example I am going to just give you say if this is the scenario first I am having these are the states, where p is true then in the next state I am going to say that AF p is true in this particular 3 state as from a first behaviour. So, these are the state where AF p is true.

Now, we are going to traverse this particular graph. And what we are going to say? Since all this particular 3 state AF p is true and this state is having only 3 successor. So, we can label it with AF p. So, like that we will go step by step and we will traverse the entire graph and finally, we will find that in all the states it will be labelled with AF p, wherever AF p is true and it will not label those states where AF p is not true.

(Refer Slide Time: 28:30)



CTL Model Checking

```
Function SAT_AF(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
end
```

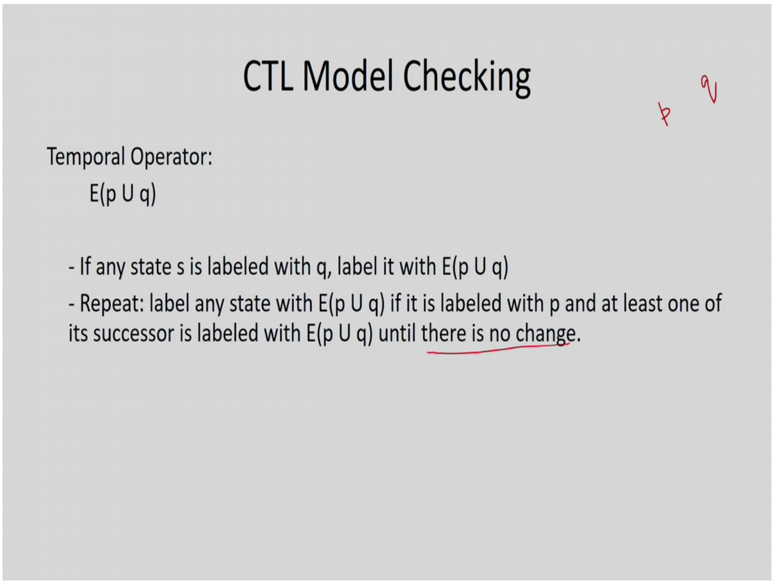
p
AFp
AFp
AFp

So, this is the test method what we are saying. So, we are starting with one particular state X is nothing but we are going to take entire state space the two local variable wising X and Y X is initial. So, initialize to entire state space. And what is Y? Y is initialised to the set of state where p is true. So, we are using this particular satisfying the function that we have defined and we know the truth values of p in a model. So, we are going to connect all the states where p is true. So, this is the state space set Y.

Now, repeat until X equal to Y. So, initially it is a entire state space and Y is the subset of the they are not equal. So, we are going to begin it. So, we are going to enter into this particular loop. Now, we make X equal to Y. So, wherever p is true we are going to put it in X for all requirement and Y, now we are going to happen Y.

So, first we know that if in a state p is true then in that particular state AF p also true. Now, we have to see we are going to connect more states s. And what we are doing? So, it is Y union we are going to connect states s; and what are what type of states we are going to connect? For all s dash with there is a transition from s to s dash where s dash belongs to Y, that means, such type of state we are going to connect say if I am having this particular scenario if these are the states where AF p is true. Then we are going to check such type of state s such that from s we are having a transition to some s dash, and where s dash belongs to this particular state Y that means, already AF p is true on this particular state. And we repeat this particular loop when X becomes equal to Y that means, we cannot get any more new state to this particular set of state Y. So, until notions we are going to repeat these things and finally, it will return Y that means, it is going to return the set of states where the given formula AF p is true. So, this is the future one.

(Refer Slide Time: 31:16)



CTL Model Checking

p q

Temporal Operator:

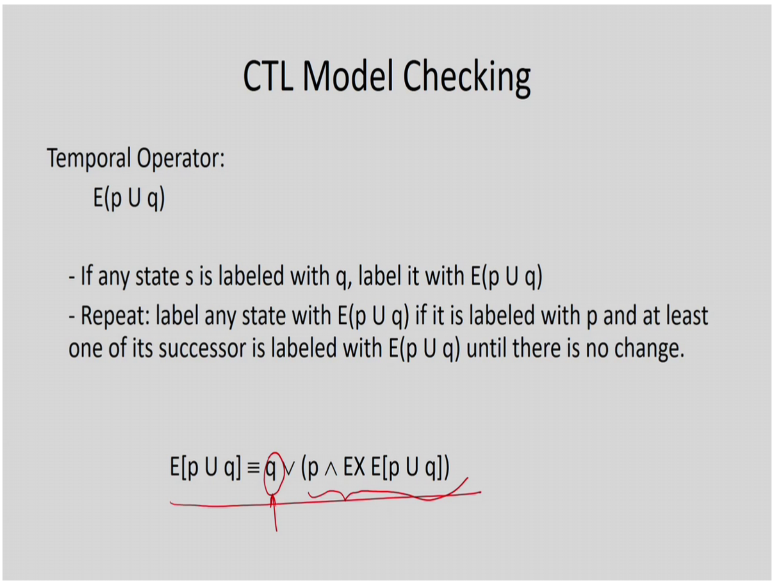
E(p U q)

- 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.

So, another temporal operator that we have is E p until q. So, there exist a path p remains true until q becomes true. So, for this you just see I am having two sub formula p and q. So, before going to look for the truth values of E p until q you must know the truth values of t as well as q that means, it is you must know the set of states where these two sub formulas are true.

So, if any state s is labelled with q then label it with E p until q, and after that you repeat label any state with E p until q if it is labelled with p and at least one of the successor is labelled with E p until q, until there is no change again this is same notion that we are going to traverse the entire state space and we are going to connect the states one after another why it is going to satisfy this particular property. And basically this is nothing but already we have seen this particular equivalence where E p until q will be true in the states if t is true itself, or it is going to satisfy this particular property p and EX E p until q.

(Refer Slide Time: 32:16)



CTL Model Checking

Temporal Operator:

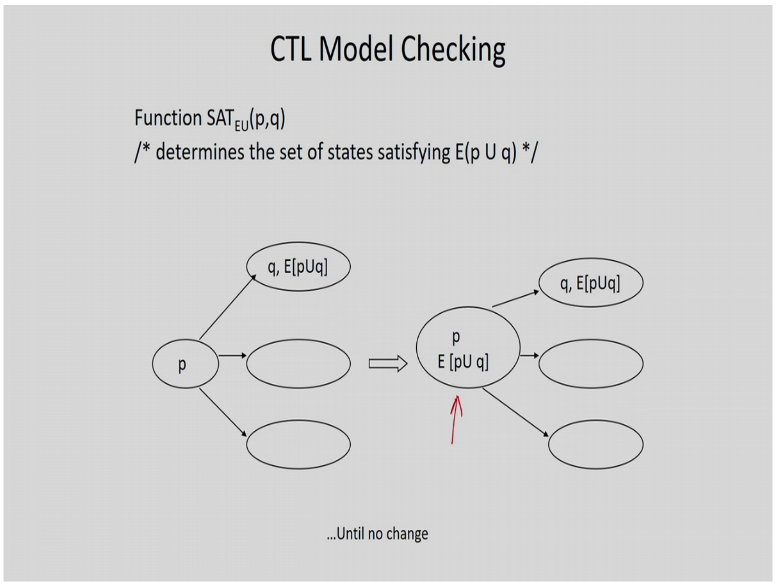
E(p U q)

- 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.

$$E[p \mathrel{U} q] \equiv q \lor (p \land EX\, E[p \mathrel{U} q])$$

So, in there exist a path in next state E p until q is true. So, this is the notion that we are having about this particular E p until q. So, if q is true in a particular state then we say that E p until q will be true. So, we are going to connect those particular states or it satisfy this particular p must be true in that particular state and there exist at least one path in such that E p until q is true in the next state this is EX p.

(Refer Slide Time: 33:11)



CTL Model Checking

Function $SAT_{EU}(p,q)$

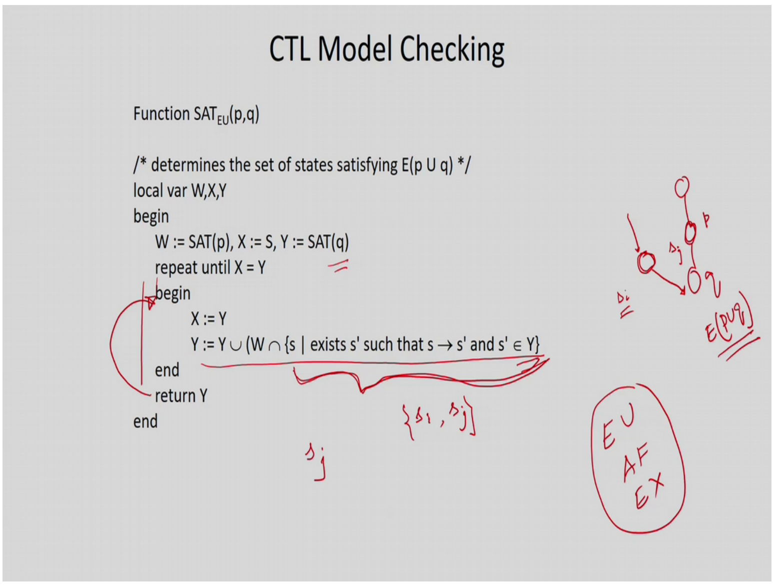
/* determines the set of states satisfying E(p U q) */

...Until no change

So, this is the scenario, if say that in this particular case if I am having this particular label then you see q is true over here, since q is true over here I can very well mark this particular state is E p until q because if p is true then E p until q is true. So, here since q is true over here, so we can label with E p until q.

Then what will happen? Now, this is the scenario then if this is labelled with your p E p until q, and q and this is p then what will happen in this particular case in this particular state p is true and p and there must exist one next state where E p until q is true. So, we are going to label this particular state also E p until q. Like that we are going to traverse the entire state space to find out the set of state where E p until q is true.

(Refer Slide Time: 34:15)



CTL Model Checking

```
Function SAT_EU(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
```

$\{s_i, s_j\}$
s_j
s_j
s_i
p
q
$E(p \cup q)$
EU
AF
EX

So, for that we are having this particular algorithm. So, what we are going to look? We are using 3 local variable W X Y. What is W? W is nothing but the set of state where p is true. So, we are connecting the set of state, where p is true because we need in our algorithm because if p is true over here then only we can look for E p until q is whether it is true in this particular state or now. So, that is why we are connecting all the set of state where p is true and X is basically initialized to in their state space because my states will grow up to that label in the formula is true in all the states. And what is Y? It is nothing but a set of state where q is true.

Now, we are getting Y set of state where q is true, and we know that wherever q is true in this particular state between E p until q is true. So, for that this means these are the initial set of state where E p until q is true. Now, we are going to repeat until X equal to Y. So, initially X is the entire state space and Y is the set of state where q is true. So, in that particular case initially it will be false then we are going into this particular look. So, for a reason X will now, assign to Y because in next iteration we are going to check whether Y remains same or we are at that some more new state space because in the next statement Y is equal to Y union something. That means, whatever Y we are having and we are going to add some new more states or if cannot add any new more state then Y remain as X, and that case it will going to satisfy X equal to Y.

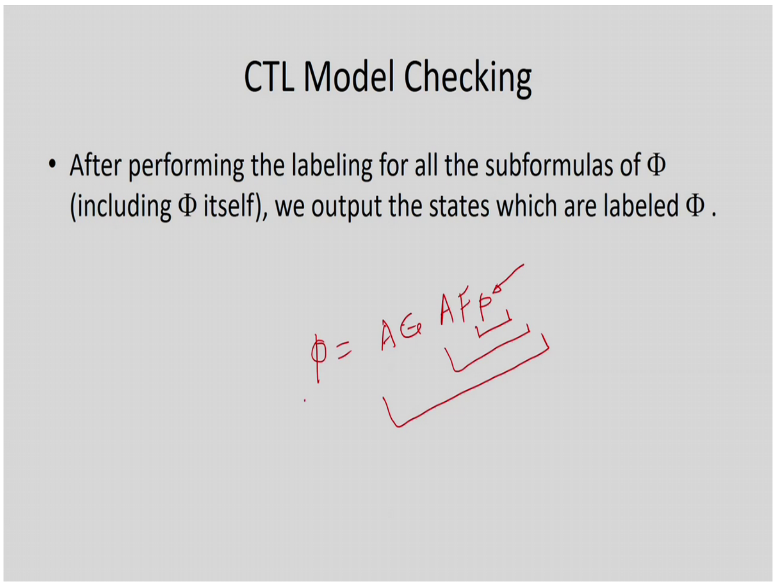
So, what are the states we are going to include? So, what is W? W is nothing but set of states where p is true. So, this particular state we are taking because p is true in the particular state then only E p until q having a change to be true in that particular state. If p is fall itself then we cannot say because if it is having something like that if q is true over here then E p until q true is over in this particular state as per our different semantics, but if p is true over here then only you can see that E p until q becomes true at that particular points.

Similarly if I am having another transition from another state and if p is not true over here then it is not a potential candidate or it is not a states where p phi p until q will be true. So, that is why we are going to look for all the states where p is true along with that we are going to see such type of states going to connect this particular states as such state there exist s dash this where we are having a transition from s to s dash, and s dash belongs to Y. What is that Y? Y is nothing but satisfaction of q or may be set of state where p until q is true.

In this scenario I am getting these two state s i and s j from, where I am having a transition to this particular state where E p until q is true, but. So, this particular component is going to give me s i and s j. Now, this subset will be having an intersection with W when I am going to intersect with W in W s i will not be present because here p is not true, but s j will be present. So, after interesting that thing we are getting s j. So, s j will be appended to Y because it is Y union something so that means, this particular state is appended. Then again it will go to this particular loop repeat and it is going to say whether X equal to Y in this case X will not be equal to Y because X is the initial state and Y we are adding some more extra (Refer Time: 38:34) So, it will be in this particular loop until we cannot add any more states to this particular set of states Y then we come out from the loop and finally, it is going to return Y.

So, these are the 3 method that we need for our model checking algorithm because other operators can be expressed with the help of these 3 particular states particular operator. So, what are the 3 operators that we have discussed? EU, then previous to it we have defined AF and EF. So, we now have method for checking the truth values of CTL formulas.

(Refer Slide Time: 39:17)



CTL Model Checking

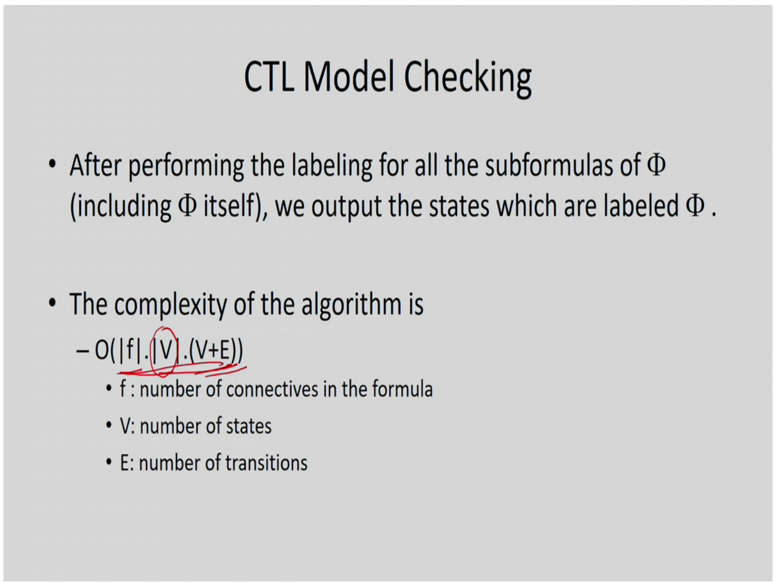
- After performing the labeling for all the subformulas of Φ (including Φ itself), we output the states which are labeled Φ .

$$\phi = \underbrace{AG\ \underbrace{AF\ \underbrace{p}}}$$

Now, after performing labelling for all the sub formulas of phi including phi itself we output the states where the phi is true. This is already we have discussed if my phi is given as your AG AF p. Now, this is one sub formula AF p is another sub formula and this is the given formula.

So, if we know the truth values of those particular sub formula then only we can find out the truth values of this particular given formula phi and in this particular case p is a atomic proposition. So, we know the set of state where p is true with our labelling function because our model is having labelling function. So, this is the way we can say and after completion of this model checking algorithm what it does? It give me the set of states where this particular formula phi is true.

(Refer Slide Time: 40:13)



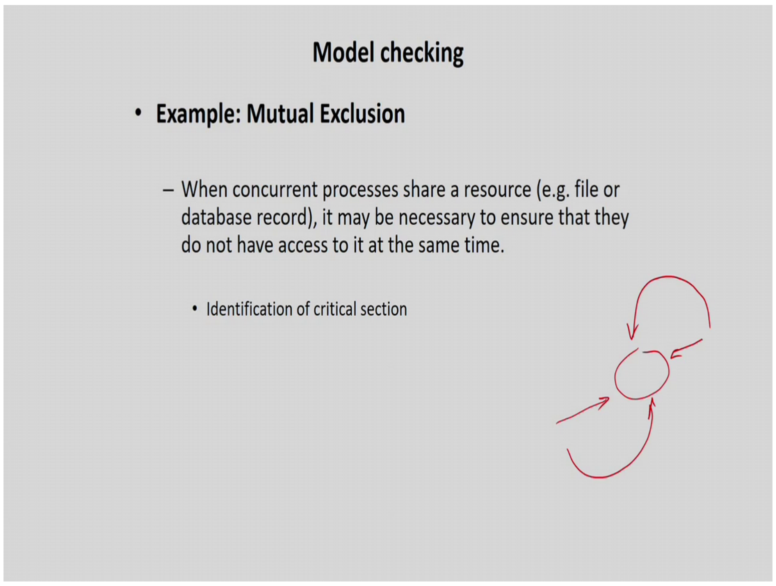
CTL Model Checking

- 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
  - $O(|f|.|V|.(V+E))$
    - f : number of connectives in the formula
    - V: number of states
    - E: number of transitions

Now, in this particular vase you just see what may be the complexity of this model checking algorithm. So, the complexity basically, so about it is the order of f V and V plus E. What is your F? F is nothing but the number of connectives in the formula, because if we are having n number of connectives then we are having n number of sub formulas including the given formula in that particular CTL formula. So, we must know the truth values of each and every sub formula then only we can look for the truth values of the given formula. So, from primitive sub formula always we need to start. So, for that many times we have to run our algorithm.

Again we are going to look for all the states that we are having, because we have to check it for all the states finally, we are going to return the set of state where the given formula is true. And we have to it is nothing but the graph traversal algorithm for every sub formula we have to traverse the entire graph and this is the graph that we are having it depends on the number of edges and number of partitions. So, this is a graph traversal algorithm and we have to traverse the entire graph for each sub formula and we have to look it for each node, each of the graph or the model. So, that is why the complexity turns up to be f V and V plus E. So, this is a algorithm and the complexities not that high. So, that is why model checking algorithm is promising to apply in our verification about system design.

(Refer Slide Time: 42:06)



Model checking

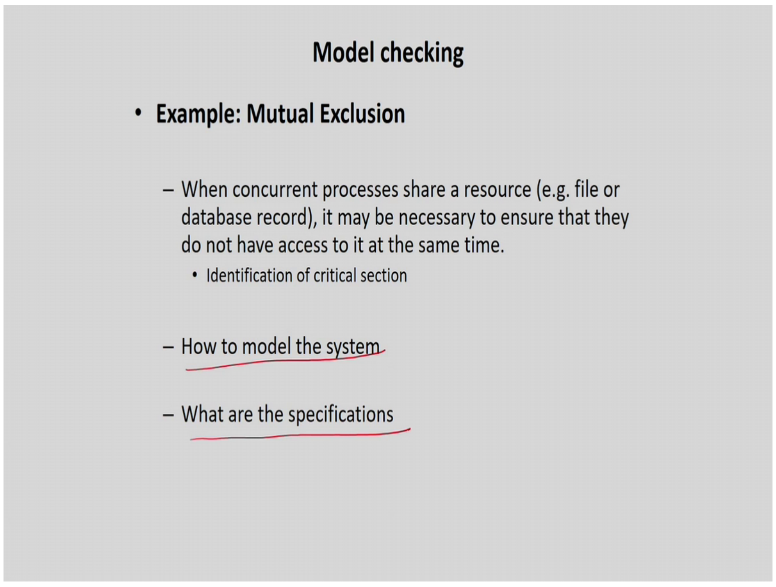
- **Example: Mutual Exclusion**
  - When concurrent processes share a resource (e.g. file or database record), it may be necessary to ensure that they do not have access to it at the same time.
    - Identification of critical section

Now, we have seen the model checking algorithm. Now, we will see how we can apply this particular model checking algorithm. So, simple example you are going to take over here which is the mutual exclusion problem that we have. It is basically an method for our resource sharing, if a particular resource is shared by many utilities then what will happen we should have the axis in such a way that it will be utilize properly and 2 or 3 processor should not disturb it content.

So, when concurrent processes share a resources for example, file or database record it may be necessary to ensure that they do not have access to it at the same time because if this is a shared resource and it is going to access by these two system then what will happen; some of the updation done by this one maybe over return by the other one. So, to achieve it, so we have to assured it. So, mutual exclusion is one of the protocol through which we can assure this thing. So, for that we are talking about the identification of critical section.

What is the critical section? If in a process a shared variable or shared resource is modified by a particular entity then we can say this is the critical section because if someone is doing some modification and others are not disturbed at that particular time. So, this critical section have to be access mutual exclusively.

(Refer Slide Time: 43:49)

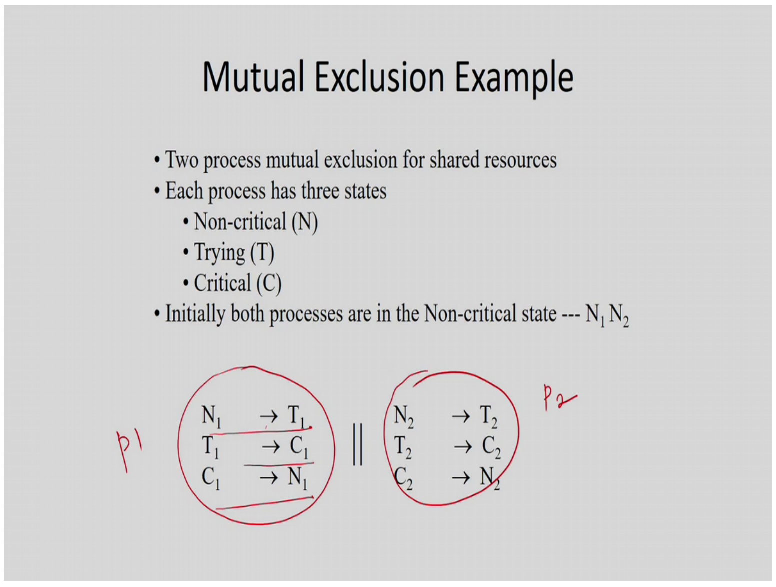


Model checking

- **Example: Mutual Exclusion**
  - When concurrent processes share a resource (e.g. file or database record), it may be necessary to ensure that they do not have access to it at the same time.
    - Identification of critical section
  - How to model the system
  - What are the specifications

So, for that we are going to see how we are going to model it. So, first one we have to see how to model the system and what are the specification or what are the properties that it needs to satisfy and how to express those particular properties. So, in that particular case already I have mentioned about a critical section where the shared resource will be modified by a particular process.

(Refer Slide Time: 44:06)



Mutual Exclusion Example

- Two process mutual exclusion for shared resources
- Each process has three states
  - Non-critical (N)
  - Trying (T)
  - Critical (C)
- Initially both processes are in the Non-critical state --- $N_1 N_2$

$$\begin{array}{lll} N_1 \to T_1 & & N_2 \to T_2 \\ T_1 \to C_1 & \| & T_2 \to C_2 \\ C_1 \to N_1 & & C_2 \to N_2 \end{array}$$

p1

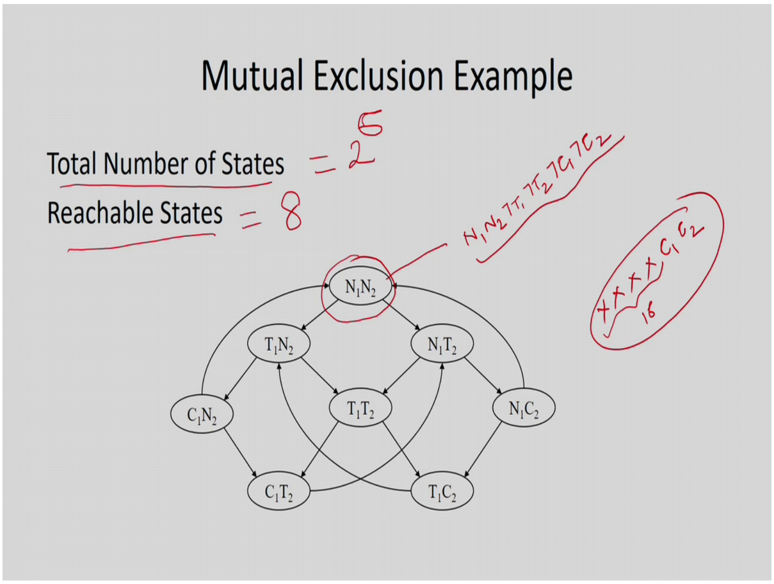
p2

When I am having a process then we can say that this process may have 3 different states. For our understanding we say that it is divided into 3 different parts. And what are these 3 parts? One is your noncritical trying, and critical. In case of noncritical it says that that means now process in a noncritical region that means, it is not doing any work with the shared resources. Trying is a process where says that, now process is trying to enter into the critical region because if I need to access the shared resource I am going to try for it but if somebody else is using it immediately I am not going to get it. So, I will just try to access it try to access this particular resource and that one I will say that process is in the trying state. And finally, the critical state once the process is accessing the shared resource and we say that it is in the critical region. So, all process can have 3 different state.

And what are the possible transition that we are having? If it is a noncritical region then always it will try to enter into the critical region if required, once it is requesting for entering in to the critical region eventually it may try get to enter into the critical region and when the system is in your critical region after compression absorb it will come out and it will be in the noncritical. So, these are the possible transition. So, these are the possible transition for process p 1 similarly these are the possible transition for the process p 2, just say that process p 1, p 2 are trying to or always access a shared resource.

Now, we have to guarantee about the mutual exclusion that both should not enter into the critical region at the same time. So, these are the 3 state we are having N 1, T 1, C 1 and these are the possible transition that means, we have a 3 possible transition for process p 1 similarly we having 3 possible transition for process p 2. Now, I have to look for the entire system. We are going to compose the entire system by these two transition system. So, one transition system is process p 1 another transition system for the process p 2 and now, we are going to compose these two transition system to get our final transition system.

(Refer Slide Time: 46:51)



Mutual Exclusion Example

Total Number of States $= 2^6$

Reachable States $= 8$

N_1N_2
T_1N_2
N_1T_2
C_1N_2
T_1T_2
N_1C_2
C_1T_2
T_1C_2
$N_1 N_2 \neg T_1 \neg T_2 \neg C_1 \neg C_2$
$\times \times \times \times C_1 C_2$
16

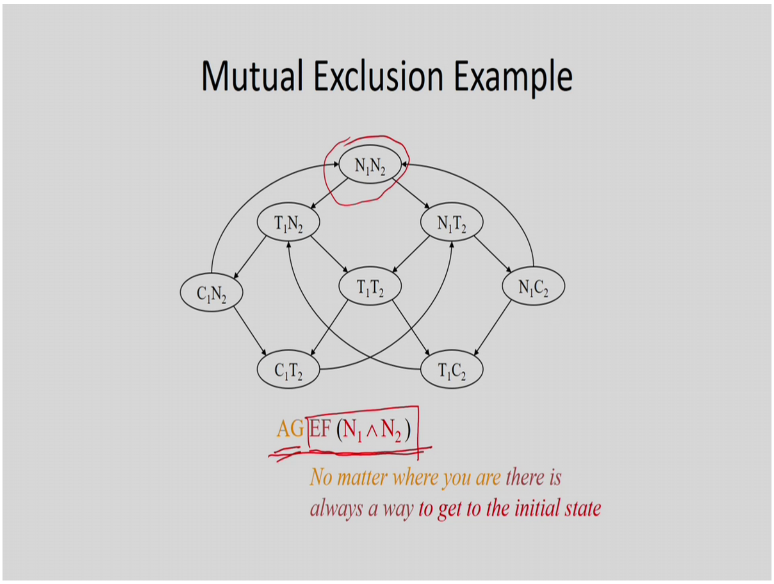
So, final transition system will turn up to be something like that. Initially both are in noncritical region so one is in your noncritical region then it may try to enter into the critical region other one is still in the noncritical region, or it may happen the process p 2 may trying to enter in the critical region and process N 1 still in a noncritical region. From such type of scenario it may happen that both of them are trying to enter into the critical region or in some cases since this trying to enter into the critical region if others one is not into the critical region then what will happen it will enter into the critical region. After completion of it job it will come back.

So, in this particular way I can say after composing this thing this is the final state space we are getting it. Here I am does writing two terms one is your total number of states and second one is reachable state. Now, in this particular case you just see that I am having 6 variable N 1, T 1, C 1 and N 2, C 2, T 2.

Now, what this N 1, N 2 means? That means, this particular state is labelled N 1 and N 2 that means, N 1 and N 2 is two over here, but the other 4 variables or state variable of all state this particular one. That means, I can say that naught of T 1, naught of T 2, naught of C 1, naught of C 2, basically is labelling is indicating like that that means, N 1 and N 2 is true over here but other 4 state variables are false at that particular point. So, total 6 state variable we are having. If we are having total 6 state variable then what is what is the possible combination. How many states we are going to have? We are going to have two to the power 6 possible states. So, this is my entire state space but in our system all may not be a valid state as per our design.

So, for that in our system we are going to say that what are the reachable state if you count it then you are going to get the reachable state is 8 only, because I am designing a for mutual exclusion. So, in a particular state I should not get that both C 1 and C 2 is true whatever may be the other 4 variable. So that means, whatever may be the other variables it may be true and false I am having total two to the power 4, 16 different combination where C 1 and C 2 is true and this is not a desired behaviour for my model for my system. So, this 16 states are not reachable. Similarly we can find out the other also. Finally, out of 2 to the power 6 we are having only 8 reachable states in our model and we are going to design about this particular model only.

(Refer Slide Time: 49:54)



Mutual Exclusion Example

$AG\,\boxed{EF\,(N_1 \wedge N_2)}$

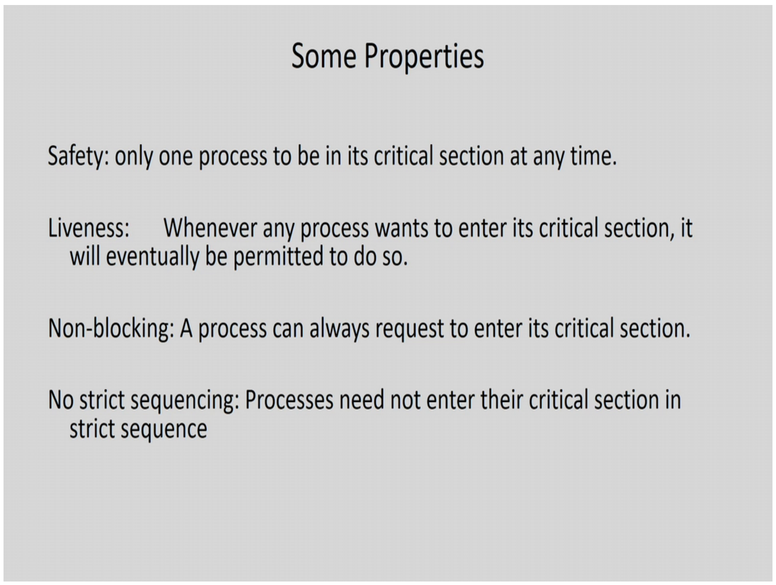
No matter where you are there is always a way to get to the initial state

Now, what are the properties that we are going to specify and we are going to verify it? One simple example just I am saying that, no matter where you are there is always a way to get to the initial state and in our system we are considering this as initial state. It say whatever may be the matter wherever you are from any position there is always a scope to come back to the initial state. So, no matters where you are, there is always a way to get to the initial state.

Now, this is basically I am saying that my system is going to satisfy this property. Now, how formally we are going to write this particular specification? We are writing this is some sort of pits English sentences but when I am going to look for the truth values or correctness of this particular property, then the if I am going to write this way it will be difficult to check for the correctness or it will be difficult to automated. So, we are going to capture this particular properties in the formal way and in this particular case we are talking about a CTL, Computational Tree Logic. So, this property can be expresses like that in AG, EF, N 1 and N 2

So, in a particular state N 1 and N 2 is true that means, this is the initial state. So, wherever you are there exist a path in future it becomes N 1 and N 2 becomes true and this property suit whole globally because you are saying that no matter where you are. So, this property must whole globally. So, that is why I am having this particular connector AG. So, such this properties specified with the help of an temporal logic formula. So, whatever property we are going to check that my system should satisfy we have to represent those particular property with the help of CTL formula.

(Refer Slide Time: 51:58)



Some Properties

Safety: only one process to be in its critical section at any time.

Liveness: Whenever any process wants to enter its critical section, it will eventually be permitted to do so.

Non-blocking: A process can always request to enter its critical section.

No strict sequencing: Processes need not enter their critical section in strict sequence

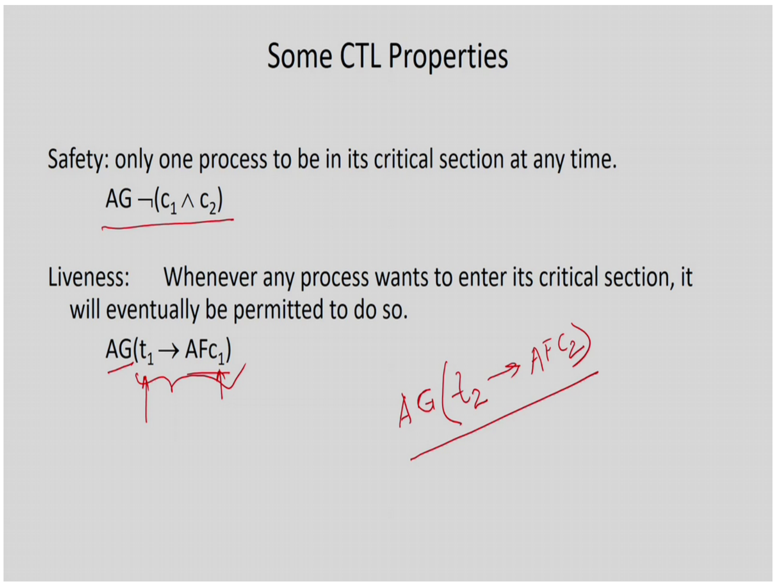
So, for such type of mutual exclusion problem some of the basic properties it should satisfy. So, what are those particular properties? First one is called safety, such safety property says only one process to be in its critical section at any time. So, this is the safety property already we said that two process should not enter into critical section with same time. So, we have to assured about the safety and it says only one process to be in its critical section at any time.

Similarly, second one is liveness, what it says? Whenever any process want to enter its critical section it will eventually be permitted to do so. So, if one particular process is trying to enter into a critical section eventually it must get a chance, if it is not getting a chance that means, we are denying the service. So, it must satisfy this liveness property.

Third one is non-blocking a process can always request to enter its critical section. So, we say it is non-blocking because we should not put any restriction that process X should not request for the critical section at some point of time. If you put such type of restriction then it is a blocking criteria. So, our mutual exclusion should satisfy this non-blocking properties also a process can always request to enter into the critical section.

And another property is no strict sequencing process need not enter their critical section in strict sequence, we should not impose any sequence just say that, a will enter after b then b will enter after c we should not put any such restrictions. So, it should be operate. So, these are the some properties that we have to always try to fulfil when we are going to design for this mutual exclusion protocol alright for. Now, this properties when we are going to apply the model checking method we have to capture those properties with the help of our CTL formula.

(Refer Slide Time: 53:56)



Some CTL Properties

Safety: only one process to be in its critical section at any time.

$AG\,\neg(c_1 \wedge c_2)$

Liveness: Whenever any process wants to enter its critical section, it will eventually be permitted to do so.

$AG(t_1 \rightarrow AFc_1)$

$AG(t_2 \rightarrow AFc_2)$

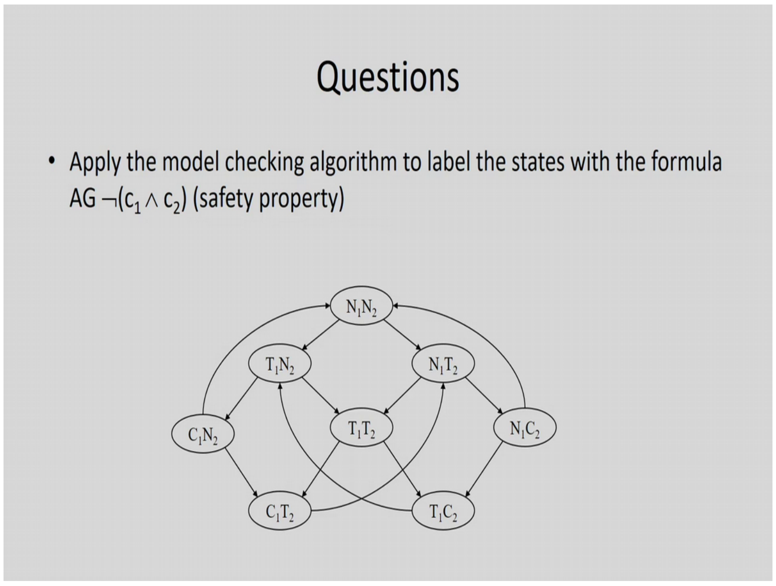
So, some of the example I am saying the safety properties only one process to be in its critical section at any time. So, how we are going to specify it with the help of CTL? This is the formula that we can write all part globally naught of C 1 and C 2, naught of C

1 and C 2 this must hold globally. So, that is why saying in all part globally naught of C 1 and C 2.

Similarly, in liveness what we are saying whenever any process want to enter its critical section it will eventually be permitted to do so. So, it says that if T 1 that means, process p 1 is trying to enter into the critical section that it implies that in all path in future AF C 1 that means, from the particular state wherever you go at least somewhere in future you should have a state where C 1 is true. So, if it is trying to enter in to the critical section eventually it should be permitted to enter into the critical section. So, this is the property and this property must hold globally. So, that is why AG. So, it talks about the process p 1.

Similarly we having process p 2. So, this is what is the liveness properties for process p 2. It is similar to this one now, we can write AG T 2 implies a f C 2. So, this is the similar property which is basically talking about in all path globally T 2 implies AF C 2 that means, a process p 2 is trying to enter into the critical section eventually it should get a permission to enter into the critical section.

(Refer Slide Time: 55:59)

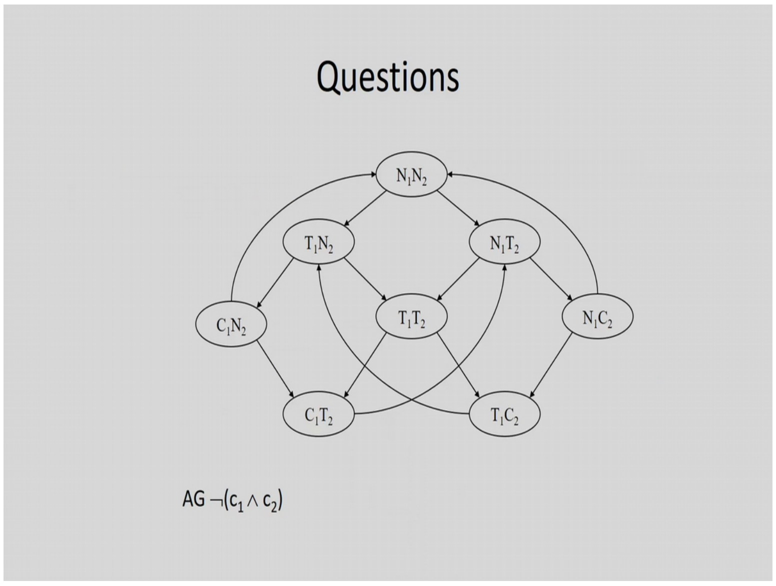


Questions

- Apply the model checking algorithm to label the states with the formula AG $\neg(c_1 \wedge c_2)$ (safety property)

Now, how to check this particular property? So, one simple property we are just going to talk about the safety property, AG naught of C 1 and C 2. So, in this particular case this is a given model already I have seen, this is in this given model we are going to look for the truth values of this particular property whether it is true in this particular given model or not.

(Refer Slide Time: 56:22)



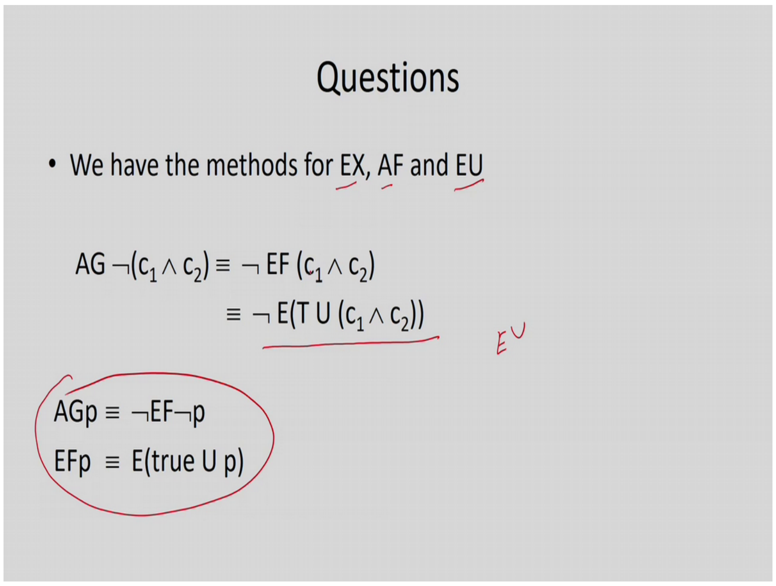
Questions

$$\begin{tikzcd}[column sep=small]
 & & N_1N_2 \arrow[dl] \arrow[dr] & & \\
 & T_1N_2 \arrow[dl] \arrow[dr] & & N_1T_2 \arrow[dl] \arrow[dr] & \\
C_1N_2 \arrow[uurr, bend left] \arrow[dr] & & T_1T_2 \arrow[dl] \arrow[dr] & & N_1C_2 \arrow[uull, bend right] \arrow[dl] \\
 & C_1T_2 \arrow[uurr, bend right] & & T_1C_2 \arrow[uull, bend left] &
\end{tikzcd}$$

$AG\neg(c_1 \wedge c_2)$

So, when I am talking about AG C 1 and C 2 I am going to find out the set of state where this particular formula is true.

(Refer Slide Time: 56:30)



Questions

- We have the methods for EX, AF and EU

$AG\neg(c_1 \wedge c_2) \equiv \neg EF\,(c_1 \wedge c_2)$

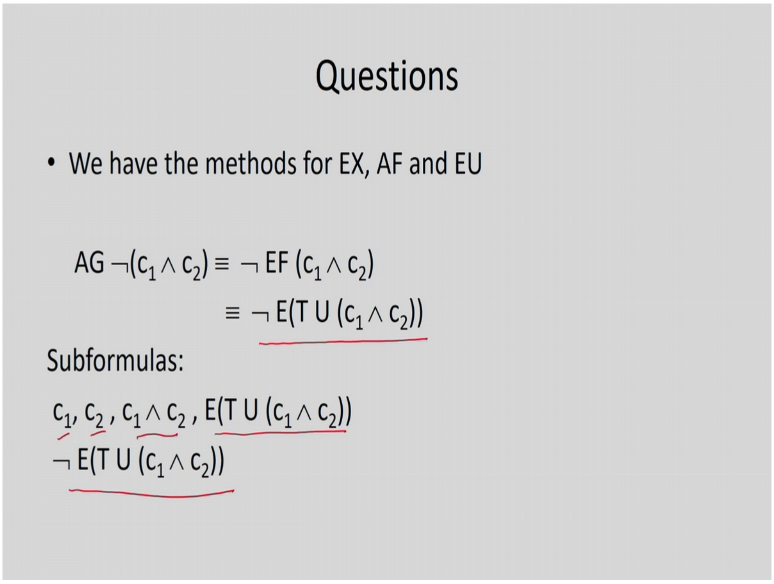
$\equiv \underline{\neg E(T\,U\,(c_1 \wedge c_2))}$ EU

$\boxed{\begin{array}{l} AGp \equiv \neg EF\neg p \\ EFp \;\equiv\; E(true\;U\;p) \end{array}}$

Now, in that particular case you just see, we do not have any operator, we do not have any method to check for the truth values of AG but we know that there is an equivalent. So, AG naught of C 1 and C 2 is equivalent to naught EF C 1 and C 2 because you have the method for EX AF and EU. So, we do not have any method for AG so that means, AG will be equivalent to naught of EF C 1 and C 2.

Again we are having a method for AF but not for EF, but EF can be represented like that E true until C 1 and C 2. Now, you just see when I am coming up to this particular formula we are having this is a atomic proposition and we are having a method for E until. So, we can apply this particular E until method to check for the truth values of this particular one. So, this is basically the equivalence sort of equivalence we have used for converting at AG naught of C 1 and C 2. So, I am mentioning over here for my ready reference.

(Refer Slide Time: 57:39)



Questions

- We have the methods for EX, AF and EU

$$AG \neg(c_1 \wedge c_2) \equiv \neg EF(c_1 \wedge c_2)$$
$$\equiv \neg \underline{E(T\, U\, (c_1 \wedge c_2))}$$

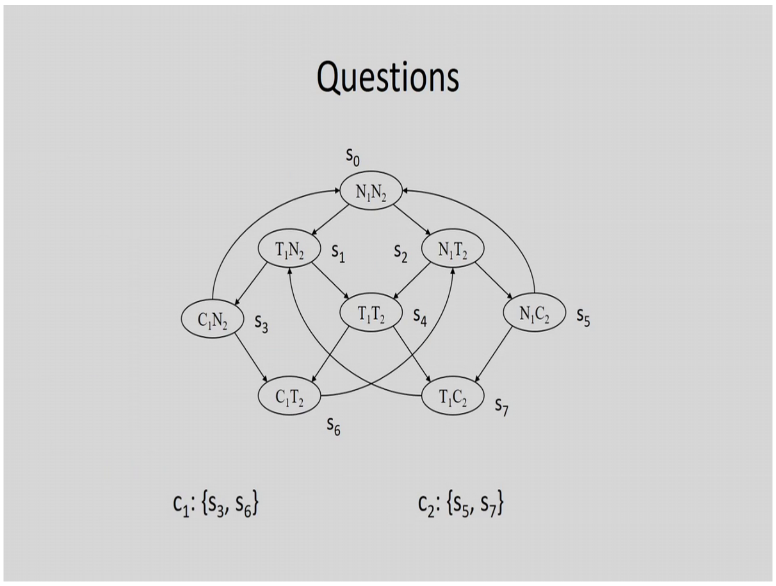
Subformulas:

$\underline{c_1}, \underline{c_2}, \underline{c_1 \wedge c_2}, \underline{E(T\, U\, (c_1 \wedge c_2))}$

$\underline{\neg E(T\, U\, (c_1 \wedge c_2))}$

Now, for this particular formula when I am going to look into it finally, we are going to check for those this particular formula. And for this particular formula you see these are the sub formulas that we are having C 1, C 2, then C 1 and C 2 that E true until C 1 and C 2 and finally, naught of E true until C 1 and C 2. Now, when I am going to look for truth values of this particular formula I must know the truth values of those particular sub formula. That means I must know the set of states where this particular sub formulas are true.

(Refer Slide Time: 58:20)

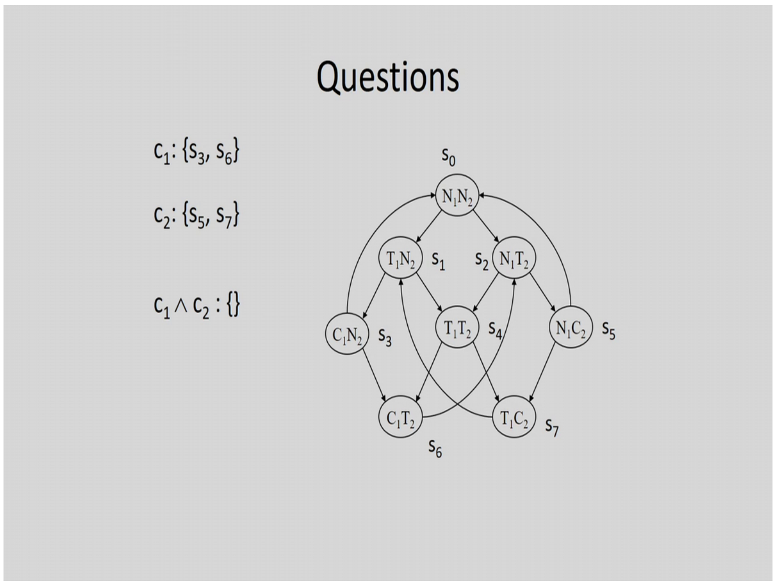


Questions

C_1: $\{s_3, s_6\}$ C_2: $\{s_5, s_7\}$

Now, in this particular case now, we are going to see that we are having total 8 step from s 0 to s 7. Now, what is the set of state where C 1 is true? So, this is your s 3 and s 6. So, in s 3 and s 6 C 1 is true this is the set of state where C 1 is true. Where C 2 is true? It is in s 5 and s 7. So, these are the two state where C 2 is true. So, we have seen we know the truth values of C 1 and C 2 or we know the set of state where C 1 and C 2 are true. So, these are my atomic proposition, and we are getting it with the help of our labelling function. So, these are the as per the labelling function we have label those particular state.

(Refer Slide Time: 59:08)



Questions

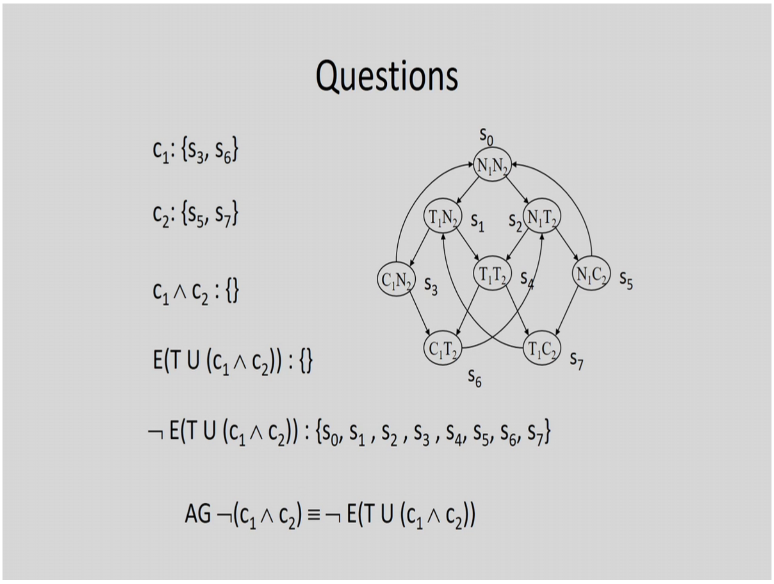
C_1: $\{s_3, s_6\}$

C_2: $\{s_5, s_7\}$

$C_1 \wedge C_2$: $\{\}$

Now, when I am having that C 1 and C 2, then what are the set of state where C 1 and C 2 are true? This is nothing but the union of these two states and ultimately I am getting phi null state, so now where C 1 and C 2 is true.

(Refer Slide Time: 59:24)



Questions

c_1: $\{s_3, s_6\}$

c_2: $\{s_5, s_7\}$

$c_1 \wedge c_2 : \{\}$

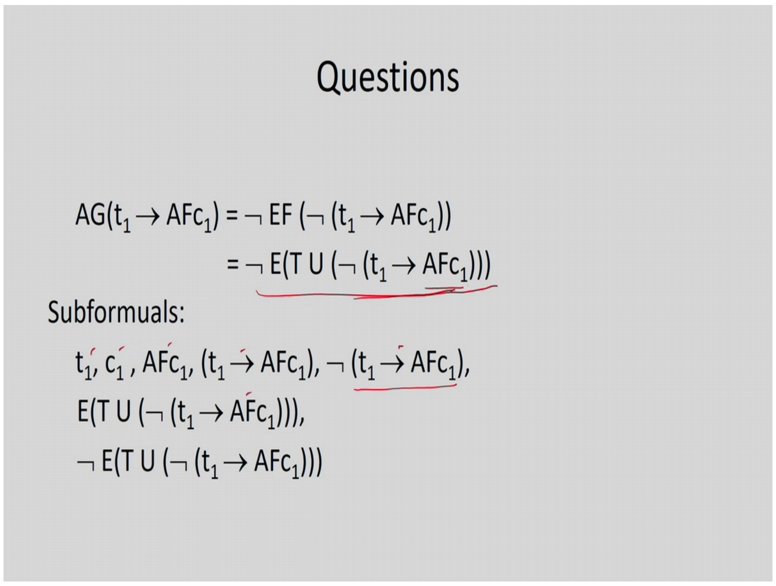
$E(T\ U\ (c_1 \wedge c_2)) : \{\}$

$\neg E(T\ U\ (c_1 \wedge c_2)) : \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$

$AG\ \neg(c_1 \wedge c_2) \equiv \neg E(T\ U\ (c_1 \wedge c_2))$

Now, next we have to see where E true until C 1 and C 2 again if you look into the algorithm then we will find them since this is phi. So, we are going to get another null state. So, in another null state we are getting, so E true until C 1 and C 2 is false in all the states. So, finally, if I take the negation of it basically what it will be there the entire state space minus this null state that means, the entire state space coming over here that means, in all the states this particular C 1 is C 2 is true or I can say that all path globally naught of C 1 and C 2 is true. So that means the given specification is valid in my given model that means, safety properties assured.

(Refer Slide Time: 60:16)



Questions

$AG(t_1 \rightarrow AFc_1) = \neg EF(\neg(t_1 \rightarrow AFc_1))$

$= \neg E(T\, U\, (\neg(t_1 \rightarrow AFc_1)))$

Subformuals:

$t_1, c_1, AFc_1, (t_1 \rightarrow AFc_1), \neg(t_1 \rightarrow AFc_1),$

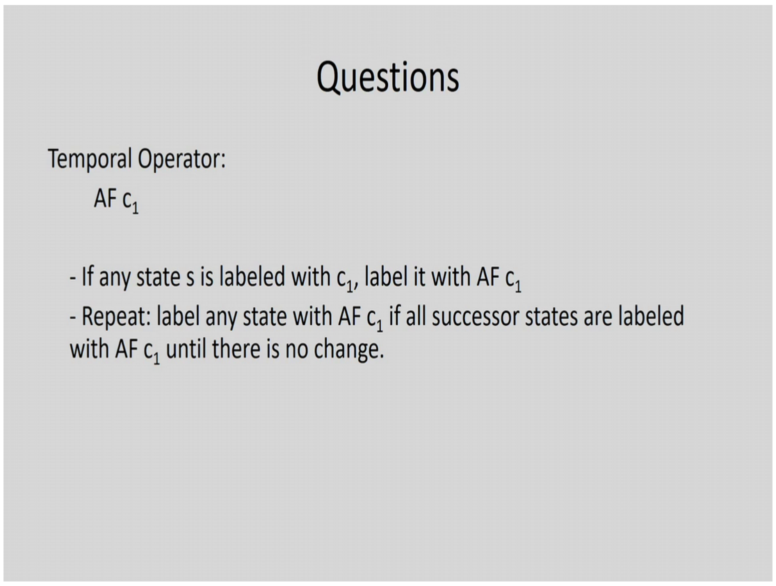
$E(T\, U\, (\neg(t_1 \rightarrow AFc_1))),$

$\neg E(T\, U\, (\neg(t_1 \rightarrow AFc_1)))$

So, similarly I can talk about the liveness property, AG T 1 implies AF C 2. Now, again I do not have any method for AG in our complex set of operator. So, finally, AG will be converted to EF and again we do not have method for EF. So, it will be converted to E t until this thing. So, finally, this is the formula we are going to look for the truth values of this particular formula in our model.

So that means, what are the sub formulas that we are having? T 1, C 1, T 1, C 1, AFC 1, then T 1 implies AFC 1 then negation of this particular, T 1 implies AFC 1, then E true until this things and finally, negation of this particular E true until naught of T 1 implies AFC 1. So, we have to look for the truth values of those particular sub formulas to get the set of state where is given formula is true.

(Refer Slide Time: 61:26)



Questions

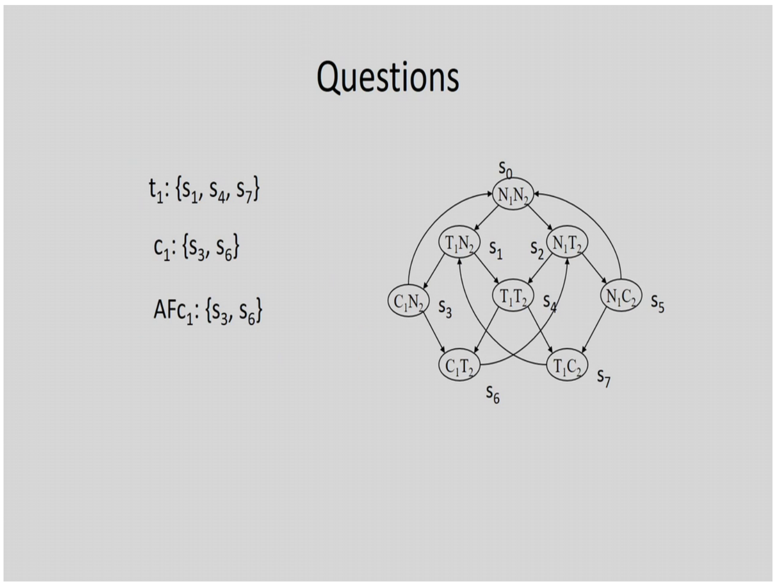
Temporal Operator:

AF c_1

- If any state s is labeled with c_1, label it with AF c_1
- Repeat: label any state with AF c_1 if all successor states are labeled with AF c_1 until there is no change.

So, already I have seen that we have to repeat this particular operator until no change.

(Refer Slide Time: 61:33)



Questions

t_1: {s_1, s_4, s_7}

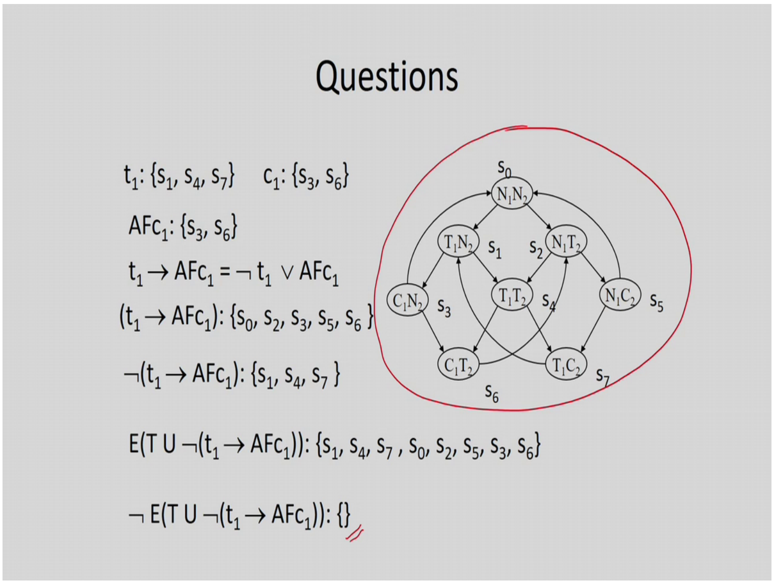
c_1: {s_3, s_6}

AFc_1: {s_3, s_6}

So now how we are going to do it? First find out the set of state where T 1 is true by looking into this things T 1 is true in your s 1, then s 4 and s 7, these are the state where T 1 is true. So, we are connecting the set of state where T 1 is true similarly we are connecting the set of state where C 1 is true.

Now, we are going to look for the set of state where AFC 1 is true. As for our semantics we know that wherever C 1 is true AFC 1 will be true in this particular state. So, s 3 and s 6 are the set of states where AFC 1 is true this is the initial point. Now, we have to traverse the particular state space to find out the set of state where this particular formula is true. Now, how we are going to say? It is in all path in future that means, we have to now, this formula will be true in all the predecessor of s 3 and s 6. If predecessor is there and if it is coming to these two states and we can say that EF may be true in this particular state.

(Refer Slide Time: 62:56)



Questions

$t_1: \{s_1, s_4, s_7\}$ $c_1: \{s_3, s_6\}$

$AFc_1: \{s_3, s_6\}$

$t_1 \rightarrow AFc_1 = \neg t_1 \vee AFc_1$

$(t_1 \rightarrow AFc_1): \{s_0, s_2, s_3, s_5, s_6\}$

$\neg(t_1 \rightarrow AFc_1): \{s_1, s_4, s_7\}$

$E(T \cup \neg(t_1 \rightarrow AFc_1)): \{s_1, s_4, s_7, s_0, s_2, s_5, s_3, s_6\}$

$\neg E(T \cup \neg(t_1 \rightarrow AFc_1)): \{\}$

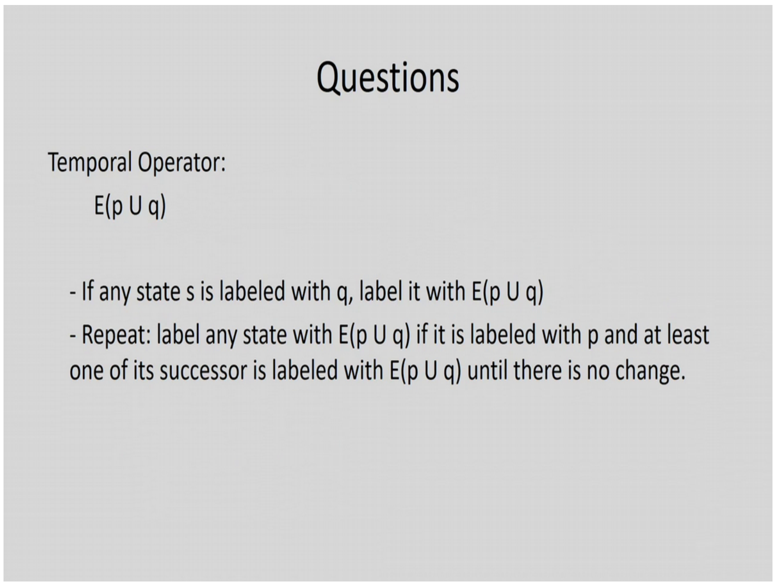
So, we are going to say that AF C 1 will be remain true in s 3 and s 6 only because we cannot connect any other state because if you say that s 3. What is the predecessor of s 3? S 1; when I am going to s 1 you just see that here AFC 1 is true. But in the other state in s 4 AFC 1 is not true. So, this will not come in to the picture.

Since, it is not coming to this is similarly if I am going to have that this particular s 6. So, what are the predecessor? One is your s 3 and s 4 s 3 is already in may but in case of s 4 in the other transition we do not have C 1. So, this will also not come. So, we are not adding any new more state to this particular states. Since we are not adding any more state, so in not to go further because there is no change. Now, in this particular states now, these are the set of state where AFC 1 is true. Now, after knowing the truth values of AFC 1 and T 1 then we can look for this particular formula T 1 implies AFC 1 and this is nothing but negation of T 1 or AFC 1.

So, now what are the states? Negation of T 1; we know the set of state where T 1 is true.

So, in case of negation the set of state where it is not true that means, apart from s 1, s 4 and s 7 all other things will come s 0, s 2, s 3, s 5 and s 6. So, these are the states where T 1 is true. So, these are the states where T 1 is not true this one union this particular state AFC 1 s 3 and s 6, s 3 and s 6 is already there. So, these are the set of states where T 1 implies AFC 1 is true. Next, we have to see what is the negation. So, negation finally, this is not there. So, I am getting s 1, s 4 and s 7.

(Refer Slide Time: 65:15)



Questions

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.

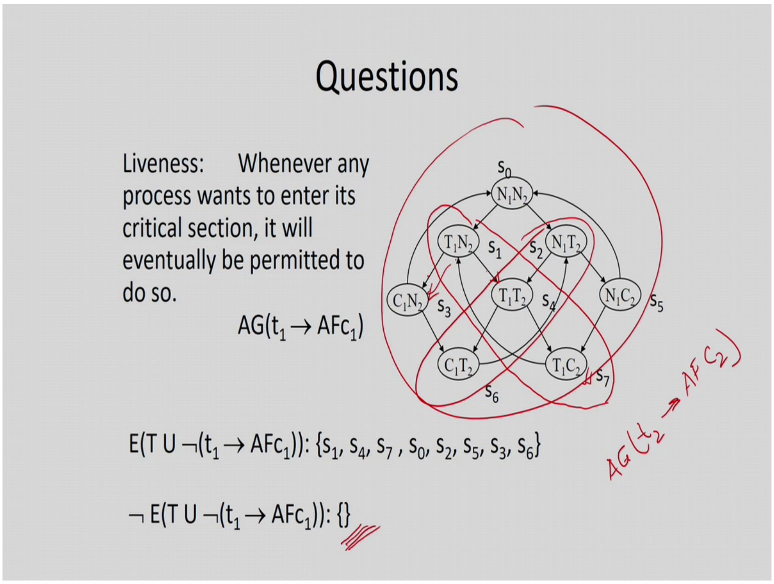
Now, E p until q this is the final operator that we are going to use. So, it says how we are doing it. So, now, we know these are the states where s 1, s 4 and s 7 naught of T 1 implies AFC 1 is true. So, E T until naught of T 1 implies AFC 1. So, in that particular case say these are the set of state where this particular sub formula is true naught of T 1 s 4 naught of T 1 implies AFC 1. So, this is the label any state s if label with q. Now, our q is that given sub formula. So, these are the set of state that I am getting initially where naught of T 1 implies AFC 1 is true.

Now, I am going to repeat this particular process, and what I am going to get considering s 1, s 4 and s 7. When I go in to the next label then we will see the predecessor of this t 3 then s 0 s 2 and s 6 will comes from s 1 I am having a predecessor s 0. So, s 0 will come into picture from s 4 we are having a predecessor s 1 and s 2 s 1 is already there. So, s 2 will come to picture and from s 7 we are having a predecessor s 4 and s 5 s 4 is already there. So, s 5 is coming into picture. So, now with this particular set of state again we are going to traverse this particular transition system then we are going to find that s 3 will come into the picture. Then finally, s 6 will also be added into the system, if I just look into the predecessor from s 0 this is the predecessor and from s 6 it is the predecessor of s 4. So, finally, these are the set of states that we are coming over here.

Now, next we have a look for the negation of this things. So, what is the negation of this things it is talking about the null state that means, it is returning a null state that means, in none of the states the given property is true. So, what it means? That means whatever I have model the liveness property is not going to ensure.

So, what is the liveness? If someone is trying to enter into the critical section it should eventually be permitted to enter into the critical section, this is the property that we are trying to check. But here, whatever model we have designed over here we checked we have seen that it is not true it is this property is not satisfying.

(Refer Slide Time: 68:08)



Questions

Liveness: Whenever any process wants to enter its critical section, it will eventually be permitted to do so.

$AG(t_1 \rightarrow AFc_1)$

$E(T\ U\ \neg(t_1 \rightarrow AFc_1))$: $\{s_1, s_4, s_7, s_0, s_2, s_5, s_3, s_6\}$

$\neg E(T\ U\ \neg(t_1 \rightarrow AFc_1))$: {}

$AG(t_2 \rightarrow AF c_2)$

s_0 N_1N_2
s_1 T_1N_2
s_2 N_1T_2
s_3 C_1N_2
s_4 T_1T_2
s_5 N_1C_2
s_6 C_1T_2
s_7 T_1C_2

So, why it is not satisfying? Now, we have to see because finally, it is giving me the empty set or null set, so it is not true. And what is not true? The liveness property. Now, since this small system now, what I can see, I can inspect it and I can find out what may be the question over here, but in model checking method when we implement it, it is giving me another benefit also when the formula is true than it is going to give me the set of states where the given formula is true, but if it is false. Now, it is turning up to be like that it is false. Then what will happen? It will give me a counter example. It says that if you follow this particular execution test then the given specification is not true. So, then we can concentrate on those particular execution test to check why it is not true and what modification I need to do in my model.

So, in this particular case what counter example it will give. So, we are saying that if it is trying into enter in to the critical section eventually it should be able to enter in to the critical section. So, if I look in to it I can inspect it but model checking method will going to give me the counter example and what counter example it will give it will give me this particular counter example. That means, if I am in status 1 it is trying to enter in to the critical section I am having a transition to s 4, from s 4 because both are trying to enter into the critical section. So, it is entering into a critical section for process T 2.

So, if it goes in this particular loop forever. Then what will happen? That, it is not process p 1 is not going to enter into the critical section because it can enter via this particular path or it can enter via this particular path. So, this is the region. So, similarly if I look into AG E 2 implies AFC 2 which is also liveness property for a process p 2, again our model checking algorithm will say that it is false and it will give me the counter example of this particular path now. So, it is T 2 is trying into the critical section but it is coming over here and this going back to this things. So, it will (Refer Time: 70:52)

So, now what will happen in this particular case? It is giving me the error that your specification is not true, it is in some condition in some situation it is going to fail. So, in that particular case I can revisit it and I can modify my model to break those two loops. So, that always it is going to assured it or there is another way we can region about it we can say that no from here I am having two different possibilities, this particular possible this manner occur always but it may eventually get a chance to come in to this particular path.

So, I may consider this particular scenario also because it may happen at when I am coming back to this particular point N 2 is now, trying to enter into the critical section then eventually N 2 will remain in N 2 process, p 2 will remain in the state N 2 and eventually it will come to this particular path and it will going to get a chance to enter into the critical section. So, I can region in this particular way and I can accept this particular model and I say that I am going to implement this particular model itself because such type of scenario is going to happen will be there.

But if I do not want to take chance then what will happen, I will relook in to my design and I will try to break these two groups. It is possible because now, I have to make some more refinement to my model so that these two loops can be broken. So, if such type of scenario is not there then finally, what will happen after breaking these two loops if I apply the model checking algorithm again in my new model, then it may happen that it will share it yes this given formula is true in your system. So, this is the way we are going to look for the formal methods to check for our specification in our design.

So, what are the things that we are having over here? You just see we are trying to design a system. So, while we are going to trying to design a system I will look for the design issues and going to have a design because my finally, my design will be some sort of finite subsystem. So, that system will be abstracted with my formal method. So, I am going to get such type of model. Once I have the model then I know the desired properties to be satisfied. I can say these are the specification of my system that I am going to design.

In that particular case look for those particular properties and those properties will be captured or can be converted to CTL formulas. Once I get the CTL representation of those particular properties then I use our model checking that things. The way we are doing over here, we give the models, and we give the CTL specification or CTL property and model checking algorithm is going to check for the correctness of those particular CTL properties. If it is true then system is going to return the truth value as true and we will be happy that, my design, a model is satisfying this particular requirement.

If it is not true that model checking algorithm find some error, then it says that no what are specification we have given it is not true. But along with that it will give me a counter example that is it will show me a execution test by going through this execution test the given property will not remain true or will not be valid.

Then as a designer what I can do? I will concentrate on this particular part and I will try to redesign that particular part the way I am saying that somehow I have to break this particular loop. So, somehow we are going to or we will redesign that particular part and try to come up with a new design. When I am coming up with a new design then again I will repeat this particular formal methods model checking techniques and eventually my design is correct, my model checking will say that yes it is true. So, once I formally check the properties in my model then we can go for the next level of my design embedded system design, then we can go for the implementation and other issues of the design. With that I am concluding today's lecture.

Thank you very much.