

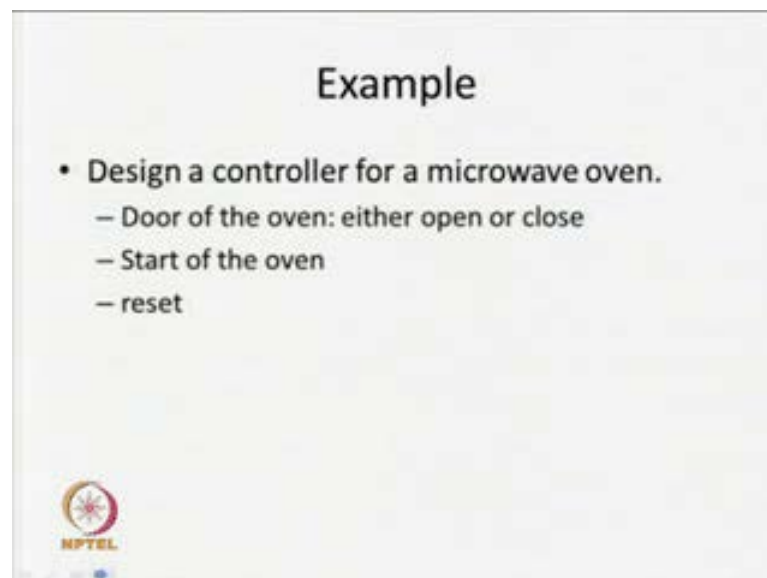
**Design Verification and Test of Digital VLSI Designs**  
**Prof. Dr. Santosh Biswas**  
**Dr. Jatindra Kumar Deka**  
**Indian Institute of Technology, Guwahati**

**Module - 5**  
**Lecture - 4**  
**Model Checking with Fairness**

We are discussing about model checking algorithm and in this modern checking algorithm what happen? We have seen that, we need one model as input and one CTL formula as another input and our model checking is look into the states where this particular formula is true. Now today we are going to see this particular model checking algorithm with some other constants, which is known as your fairness constant.

Why we have looking into the fairness constant? Why it is required? For that we will just see our general model checking algorithm with an example. We will try to devise an example; we will go through the example. And later on I will say why this fairness is coming in to the fixture and how it is going to help us during module checking.

(Refer Slide Time: 01:09)



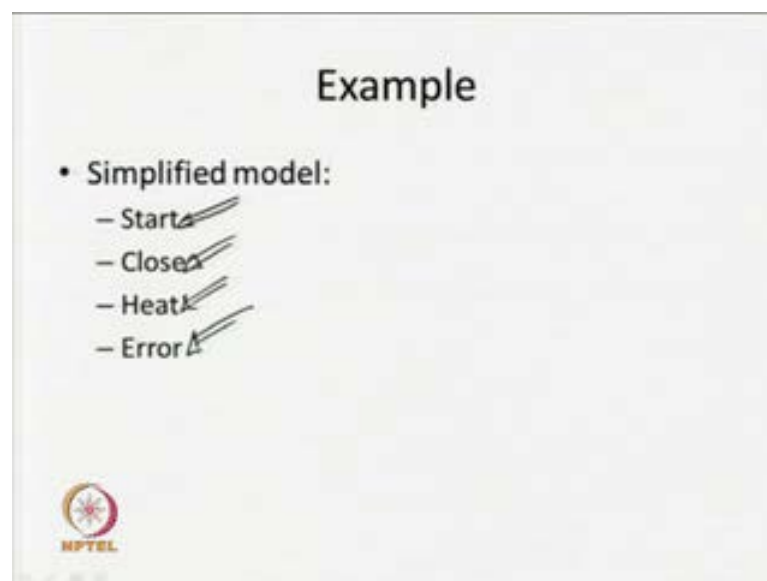
Now just say, first we are going to look a design issues. That one example; design a controller for a microwave oven. All of you known that, microwave oven is use for cooking food; so it is automatic. So it should have a controller to control the operation of this particular micro oven. Now when we are going to design a controller, initially we are

going to look for an abstract model of that particular controller. And depending on that particular abstract model, we will design it later on refining and we will go for more details.

Now, when you look into it done, what is the requirement for this particular controller? We know that there is a node for your microwave oven. We can either open the door to put a food item inside the oven and we can close the door; that means, we need some mechanism and generally it is a sensor, which will send the signals sensor either door is open or door is close.

Let on after putting the food items and after closing the door we can start the oven. When we start the oven, then what will happen? The heating coil will heat up and eventually the food will be cooked. So, by looking into all those issues, we can come up with your very simplified model initially.

(Refer Slide Time: 02:30)

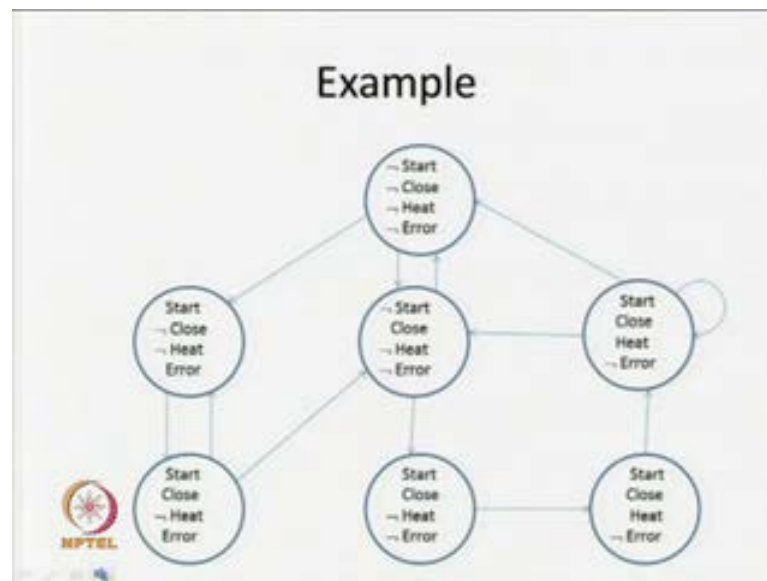


Simplified abstraction and we will find that some of the things or some of the signals required to model this particular controller. So, those signals are basically we acquire abstracting out and we are saying that first one is start. It is going to say that the oven has been started. Now it is going to flow up cooking have a food item. Close basically indicates that whether the door is closed and the door is open. One we start that heating coil will be heated up and food will be cooked. So we can say that another important issue, that we need is

Heat and sometimes they oven may go into the error situation, so you can call take another signals call Errors.

So, with the help of these four signals, we are trying to design our controller. So, this is some set of abstraction later on we may have to refine it and we can go for a complete result. But, first we will see this particular obstruct model and we will see how we are going to look for a property that has to be satisfy this particular model.

(Refer Slide Time: 03:28)



Now, with these particular things now, what we see that? Now, you have to loop for the states, so basically we can see the operation of our micro oven. Just see first we are saying that this is a state I can marked it as a s 1. I am saying that knot of Start, knot of Close, knot of Heat, knot of Error. That means the microwave oven is in ideal situation, it was not started door is not close mean door is open. We can say it is not heater up and there is no error combination. Now, these are the x on that, I can say that; I can close the door and it is coming to this particular state whether not start, close, heat and error. Because, it was door was open now door is close.

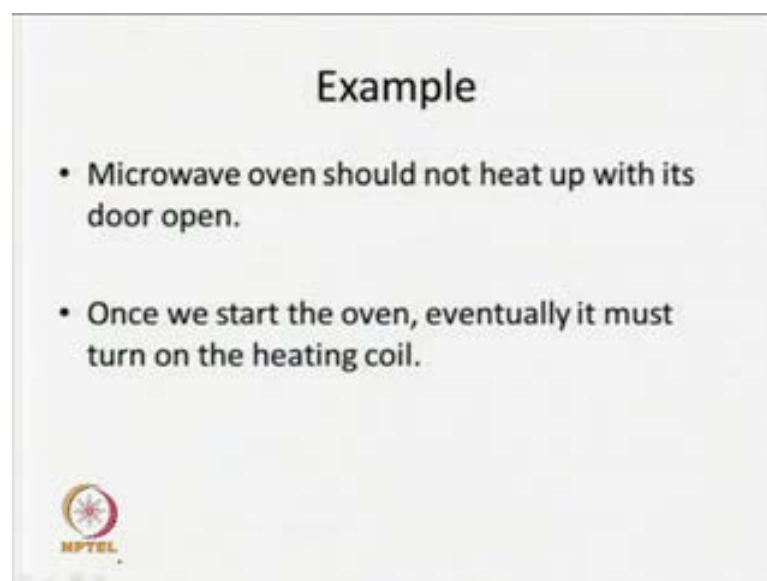
Now, one close the door it again an open it; because he may up to would some items, so he can be in this particular lower. He can close the door, he can open the door. But sometimes it may happen, that we are in that ideal situation either condition; now somebody has start the oven. Now it is not close, that means door is open. So, in this particular case he has started the oven, then now what will happen? This is Start

becoming true and it has gone to the Error condition. Now when it was gone to the error condition? Now, usual might have realized and he is going to close the door but, again it is still in the error condition. He will open the door he may be in this particular look trying to open. So we have given an option reset button, through reset he can come down to this particular time.

Now, say my door is close; it is not in error condition. Once we have coming to this thing, then we can start the ovens. So start is true, close and not it not. Then we can go for a warm up, that means you are starting the heat so heating coil is ring up; then will start cooking. So it will remain over here and it will one is completes, then coming to this particular thing. That door is closed and we can open the door or straighter yopu can open the door and come to this.


So, this is the simplified operation of our micro oven. So now in this particular case, now we see that, when we go for model checking, we need a Kripke structure. What is a Kripke structure? It is being set of states, we are having transition and we having labeling function. So, this particular events are not required for our model checking that start oven, close door, he set warm up; all those things are not equal but, we need the other in function are those steps this labeling function and transition.

(Refer Slide Time: 06:05)



**Example**

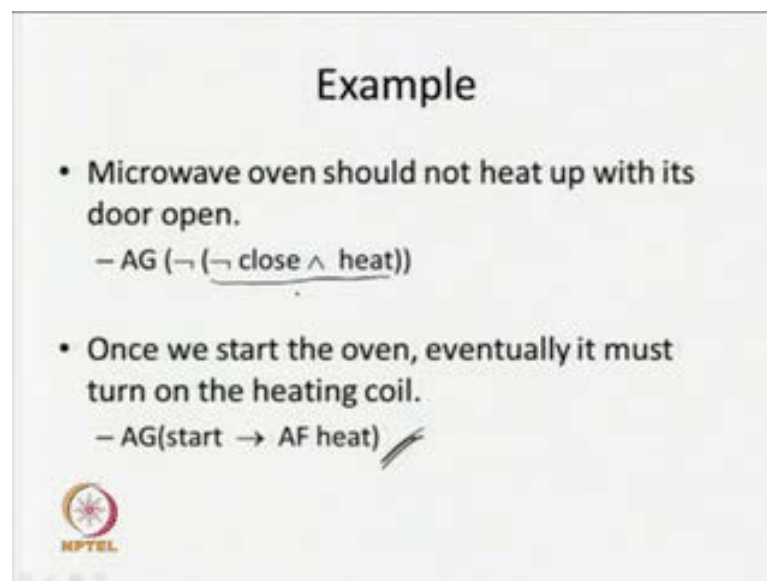
- Microwave oven should not heat up with its door open.
- Once we start the oven, eventually it must turn on the heating coil.

  
NPTEL

So, basically what happens? You can say that eventually you are come up with this particular model and this is a simplified model; and this particular model and in this Kripke structure we are going to check some of our properties.


Now, what are the properties that it should satisfy? Say I am just coming with two examples; first one I am saying that, microwave oven should not heat up with its door open. So when is door open, so coil should not get heater up; because still we are going to some (( )) Second property we are saying once we start out when eventually it must turn on the heating coils. So, once we starts the oven it should go for cooking mode, so heating coil must be heater up, so these are the two properties. Now how we are going to check these things, so we know that we are looking for CTL model checking, so somehow here to represent these two properties in CTL formulas.

(Refer Slide Time: 06:41)



**Example**

- Microwave oven should not heat up with its door open.  
–  $AG(\neg(\neg \text{close} \wedge \text{heat}))$
- Once we start the oven, eventually it must turn on the heating coil.  
–  $AG(\text{start} \rightarrow AF \text{heat})$

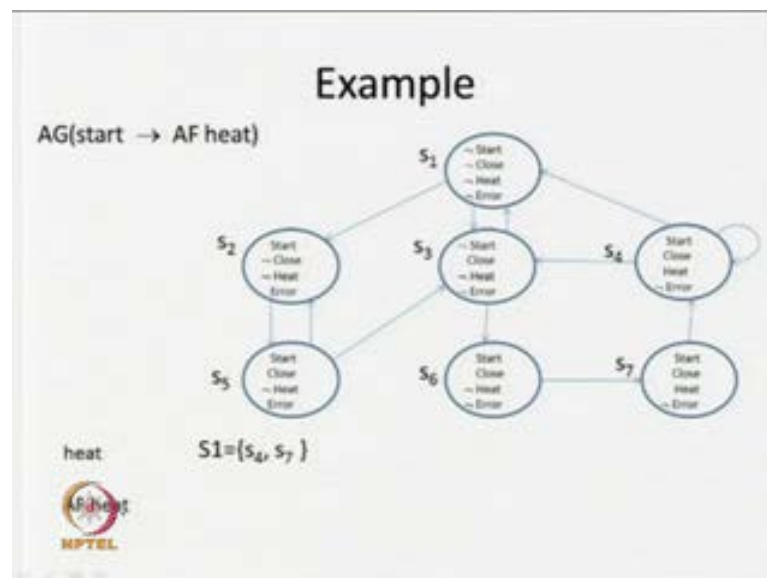
 NPTEL

So, the first one is very simple so basically knot of close and knot of heat should not be true together. So, that is why I saying that knot of close and heat. So, these two combinations knot of close that means, door open and heat it should not be true together. So negation in front of the we are putting that negation. And it should whole globally, so in all path globally these properties must whole. Second one I am saying that we start the oven eventually it must on on the heating coil. That means I am turning on the heating coil, that means oven will go in to the heating condition. So, I am saying that in all path

globally start implies, A have been all path in future is here; heat is true. So, I am capturing this particular property, with help of this particular CTL property.

Now, I have these two CTL properties according to my requirement and we are going to check whether these two properties are true or not. If you look into the first property it is simply based on the automatic proposition; I having one automatic proposition close and another automatic proposition heat. So we have to check whether in which situation, in which state there are true or they are false depending other I can look for the A G. And second one I am having one start and heat again A F and A G; A G is involve and older. So I am going to check property for a second, first one will be simple one and you can check it yourself; will see how I am going to check this particular second property.

(Refer Slide Time: 08:12)



Now we are having this particular Kripke structure or model about system, microwave controller and I told you that you are going to check for the correctness of this formula in all path globally start implies A F heat. So first we are going to see that we have to look for each and every sub formula, first we are going to look for this particular sub formula heat. Now, by looking into this Kripke structure we will find that heat is true in state  $s_4$  and  $s_7$ . So this is  $s_4$  and this is  $s_7$ , this is heat is true in procedure. Now, by looking into these things since it is true, now we have to go for, we know which are the states where heat is true, so we are getting  $s_4$  and  $s_7$ . Then we will go for the next sub formula


A F heat; so A F heat. Now you have to how we are going to look for the correctness of this particular A F heat.

(Refer Slide Time: 09:01)

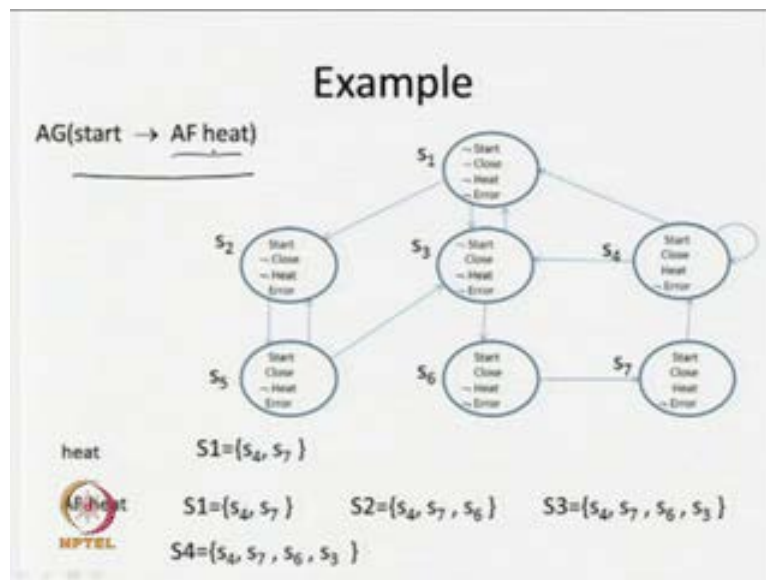
### Examples

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.



(Refer Slide Time: 09:31)



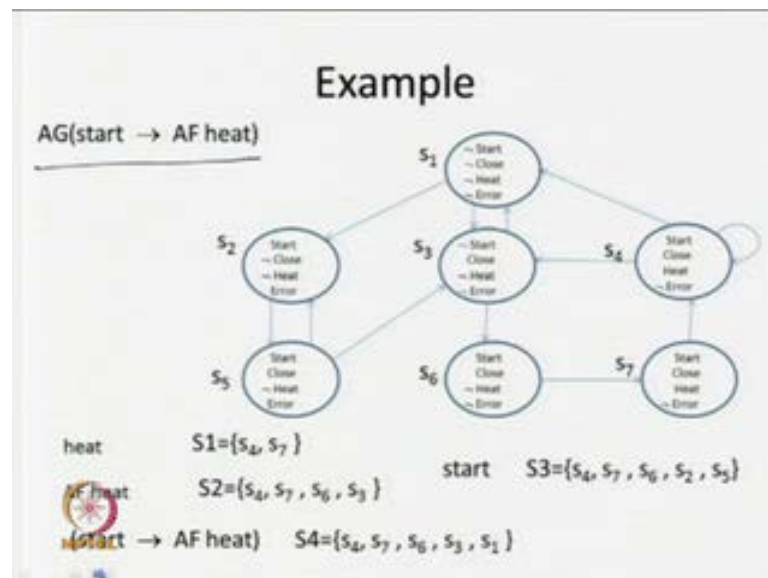
So, how we are going to do? We know this temporal operator A F  $c_1$ . Now what will happen? Now when we state  $s$  is labeled with  $c_1$  label it with A F  $c_1$ . This is the first step then we are going to repeat this procedural. Label any state with A F  $c_1$ , if all successor states are the labeled with A F  $c_1$  until there is no change. So, we know this

particular state, now we are going to apply this algorithm to look for a states where A F heat is true.

So now in this particular case, so we now the A F heat is true in a s 4 and s 7. Because heat is true in these two states, so this is our first step. Now, after that we will go into the particular repeat step. Now what will happen? Next we are going to see where it will be true, so s 4 and s 7 we are getting it. Now we are going to see, what are the predecessor states of these particular two steps? So s 4 it is having its own predecessor and s 7 is having s 6 as a predecessor. Now only if it is now, what will happen? We are going to label a state, if all of its successors are labeled with A F heat. Since it is having one successor and or it is labeled with you're a F heat, s 6 will also be labeled with A F heat. So in A F heat will be true now s 4, s7 and s 6.

Now, since there is a sense of state so we are going to repeat this particular process. If you look it to don s 6, we are going to get this s 3 predecessors and in this particular case, it is 7; this s 3 having this particular successor and this is a scenario of that we having, now what will happen? It is having this particular predecessor s 3, now we are going to see, all of it successor.

(Refer Slide Time: 11:47)



Now what are it successor? One is your s 6 and second one is your s 1. So in case of s 3 you will find that it is having this particular successor, so s 3 will be labeled with your with this particular A F heat. Next will go and we will find that remain is you s 3,



because when I am coming to this particular s 3. So it is having this predecessor your s 1 and it is having this predecessor your s 5.

But, they are not labeled with your A F heat, so it cannot include s 1 and s 3, s 5. So our remains that remains a s 4, s 7, s 6 and s 3; so these are the state where A F heat is true. Now once we know the states where A F heat is true, now we can go for this particular sub formula start implies A F heat. That means if start is true, then A F heat must be true. So, in this particular case how I am going to say that, these are the state s 4, s 7, s 6, s 3 and s 5. These are the state I am getting, because it is knot of start or A F heat. So we know that A F heat is true for s 4, s 7, s 6, s 3; these are the states and along with a knot of start is true over here, so knot of start because we know that p implies q is your knot of p or q.

So, in these particular four states, your A F is true and knot of start is true in s 1; so s 5 will also come into your this particular scenario. So ultimately we are going to get these are the states where start is true and eventually we are going to get A F start implies A F is true in your s 4, s 7, s 6, s 3 and s 1. So I am going to get this particular states.

(Refer Slide Time: 13:01)



**Labeling algorithm for AGp**

AG(start  $\rightarrow$  AF heat)

- Step1: Label all the states with AGp.
- Step2: If any state s is not labeled with p, delete the label AGp.
- Step3: Repeat: delete the label AGp from any state if all of its successors are not labeled with AGp until there is no change.

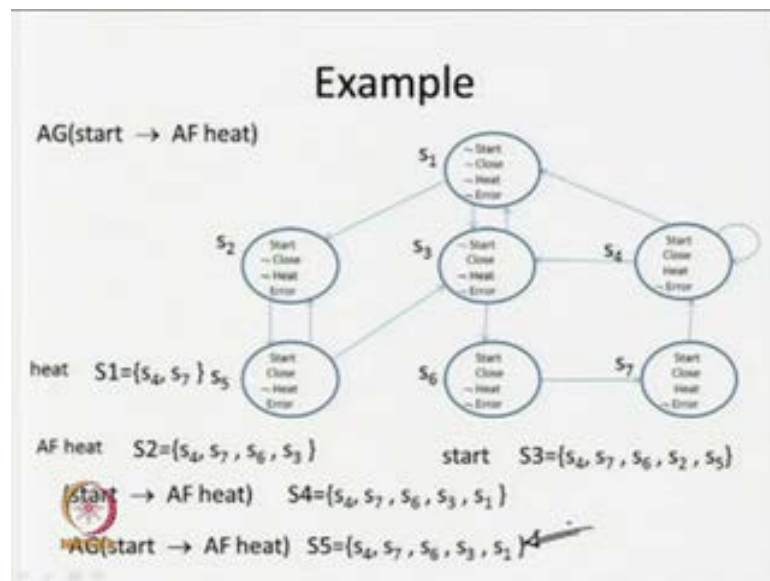


Now, after that once we know the labeling of this particular sub formula, then we can go for mains of formula. So, in these particular mains of formula what will happen? Now we are having this particular A G start implies A F heat. We know the procedure, now what is the procedure of A G? So, how we are going to get? So step one label all the

states with A F p, then if any state s is not labeled with p, delete label A G p. so, basically initially we have going to say, where A G p is true then will repeat this particular steps. Delete the level A G p from any state, if all of its successors are not labeled with A G p until there is no change.

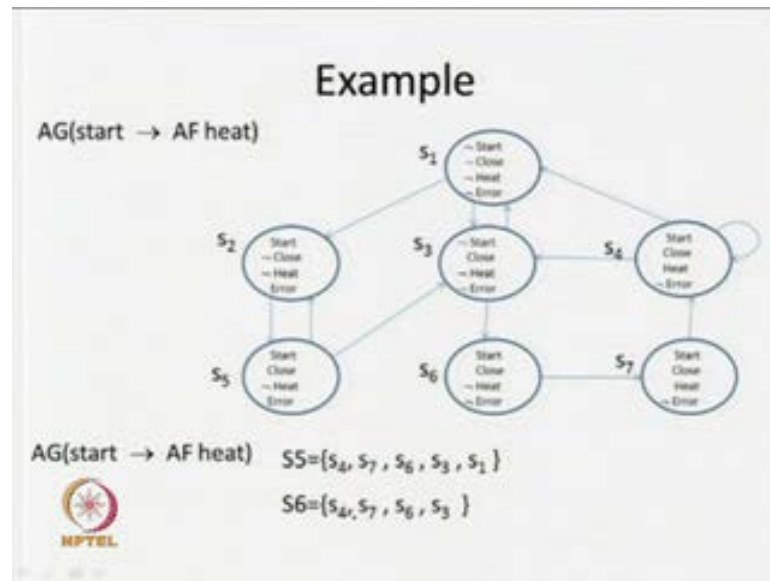
So, we have going to repeat this step three. So, if you look into procedural then what will happen? For A F start initially we have going to label, the all states then will removed where it is not label with your start implies A F heats. So, these are the steps where start implies A F heat is true. So, A G starts A F heat will be true in your s 4, s 7, s 6, s 3 and s 1. In this particular five states it will be true these two steps will not come. Now we are going to repeat this particular step there, delete the label A G p from any state if all of its successor are not labeled with A G p until there is no change.

(Refer Slide Time: 13:30)

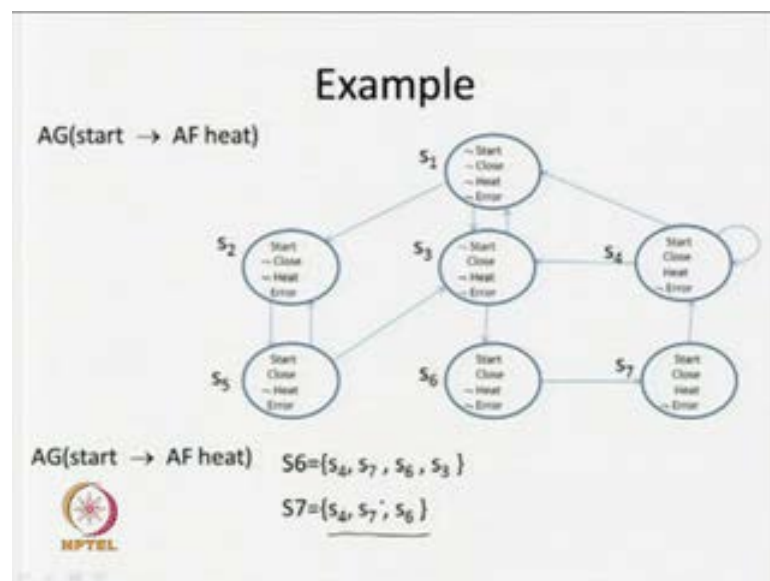


So we are starting with this particular state s 4, s 7, s 6, s 3 and s 1. Now we are going to repeat this step and see what are the states will eventually remain over here. Now in this particular case, you just see that; s 1 it is having successor s 3 and it is having successor s 2. Since all the successors are not labeled with start implies A F heat, so we have to remove s 1 from this heat. So eventually I am getting these states s 4, s 7, s 6, s 3. So I am going in once inside the loop.

(Refer Slide Time: 14:19)



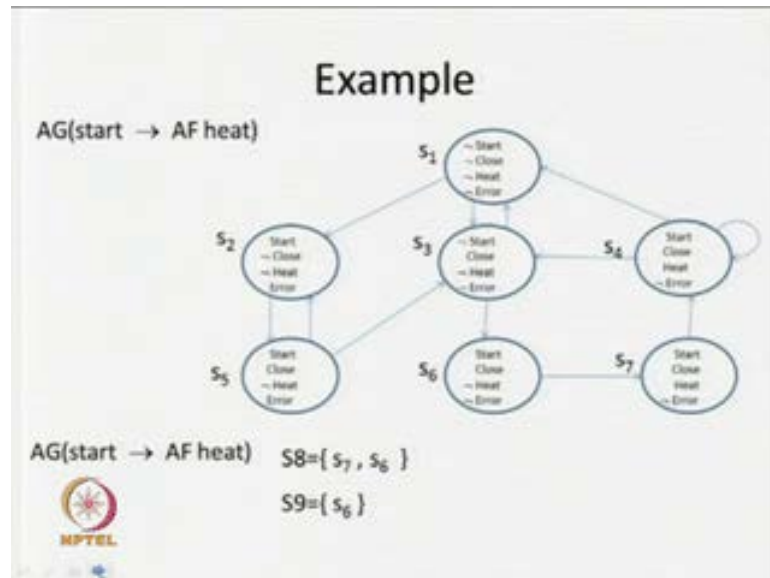
(Refer Slide Time: 14:45)



Now will see, again since there is a sense we have to go again inside this particular loop and will find that now when I am coming to this particular state s 3, now how many successor it has. It is having successor as s 1 and it is having successor s 6. Now s 1 is not labeled with A G start implies A F heat. So we are going to remove s 3 even. So we are going to get s 4, s 7, s 6 as the remaining state. Now, from these particular things now we look again these particular conditions. And what will happen?

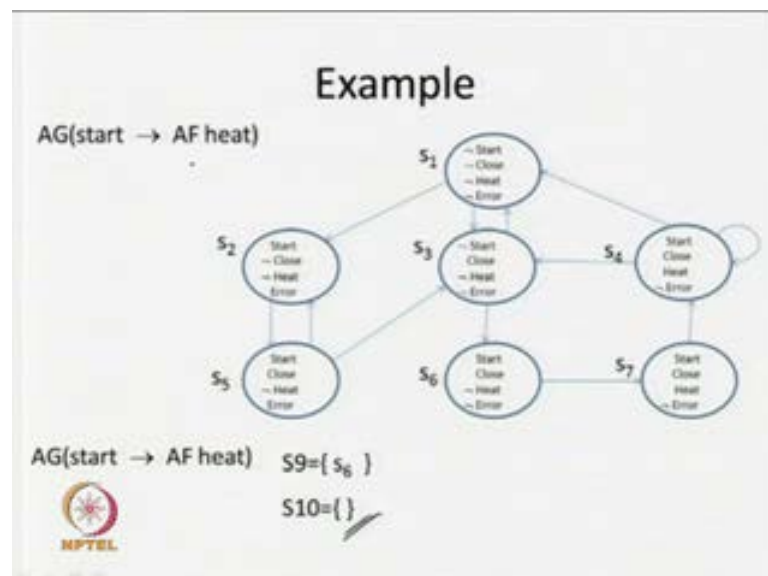
In  $s_4$  now you just see that, it is having  $s_4$  has a as one successor. It is having one successor and  $s_3$  and another successor is  $s_1$ . But this A G start implies F it is not true in  $s_3$  and  $s_1$ , so we are going to remove  $s_4$  also. So ultimately we are remaining with  $s_7$  and  $s_6$ .

(Refer Slide Time: 15:37)



Now, when we come to this particular state; now when you come to  $s_7$ , it is having one successor  $s_6$  and it is not labeled with this particular formula, so will remain  $s_7$  from here.

(Refer Slide Time: 15:49)

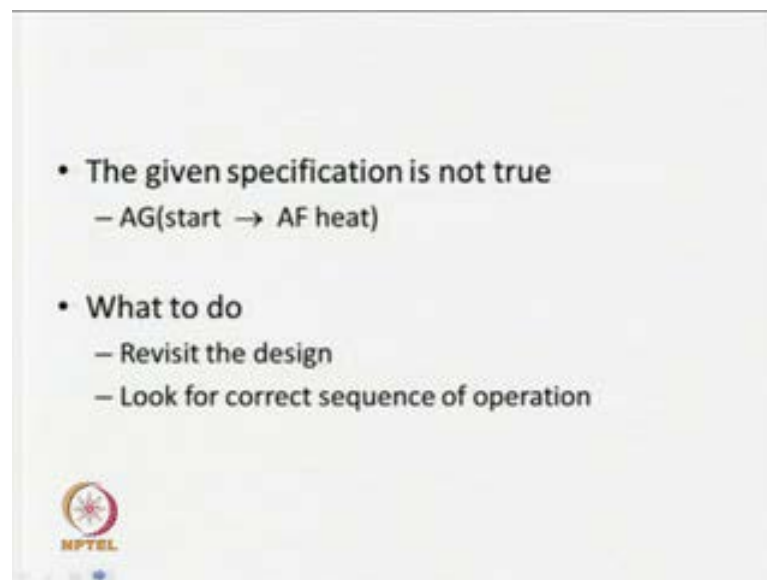


Now when I am coming to s 6 then will find that it is having one successor s 7, it is not labeled with A G start implies A F heat; so we remove that one also and eventually we are getting an completion.

That means in none of the state this particular formula s 2, so that means A G start implies A F heat is not true in this particular model. So now what we have going to do? Now you say that, we have come with a model and you are looking to check for a particular formula. And eventually we will find that after going to a model checking algorithm we found that this particular formula is not true. Now in this scenario what we have to do?

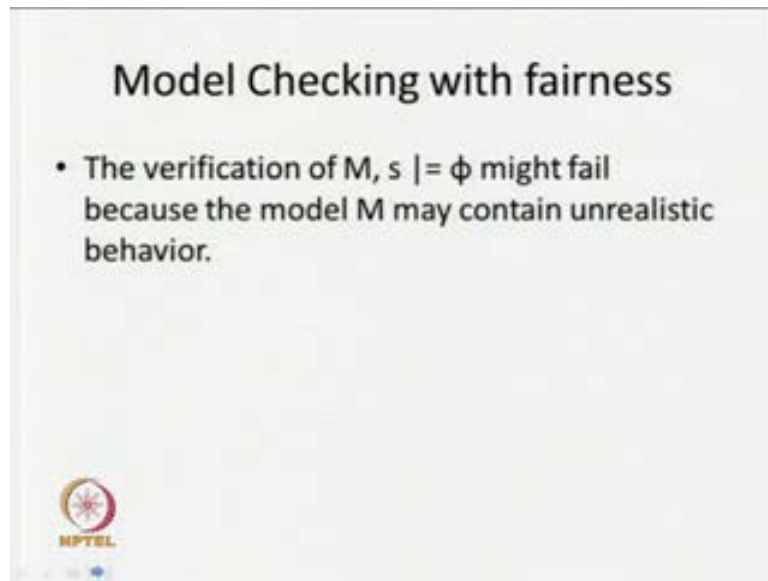
We have to revisit our design and try to modify the design, in such way that the design formula a given property will be true in our model. But if you look into this particular model up to some extent will find that this model or this design somehow correct. Because, the design I will convince it, but my model checking algorithm is saying that it is not correct. Now either I have to revisit it or whether can I do which something else. Now we are going to look in to that particular issue.

(Refer Slide Time: 17:01)



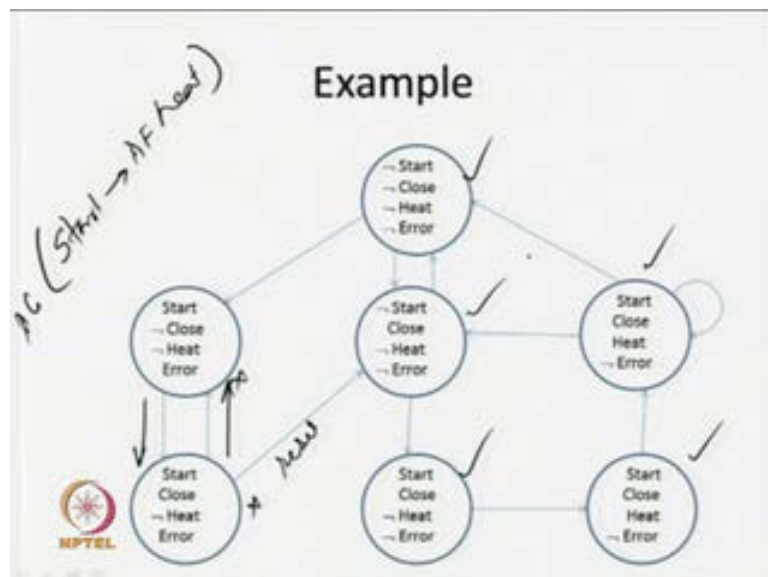
Now either we can revisit our design or we can look for a correct sequence of our operation.

(Refer Slide Time: 17:11)



It may happen that when we are going to look these things at, when we are going to look our verification of properties in our model. It may happen that we are having some unrealistic behaviors. Due to that unrealistic behavior, it may happen that the given property is not true in our model. So some of what we are going to do? We are going to filter out this particular unrealistic behavior.

(Refer Slide Time: 17:36)



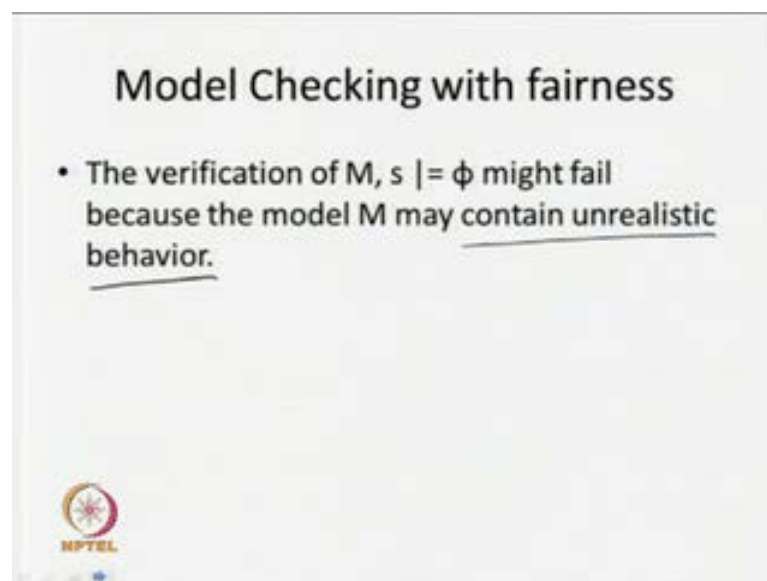
Now just see this particular example, if you come up with this particular model. Now if you find that, if you look for these. Say we are saying that start implies A F heat. If you

have started the heater then eventually on path, in all path in future it should be heated up.

Now you have seen that these particular formula is true in this particular five states. These true in this particular and when but it is not true in these two states. So that is why when I am going to look for A G of this total things due to this particular states it is not true in our model. But what is happening over here? You just see that, you know that this is your starting with then you are closing the door since it is not going to heated up again we have opening the door.

So closing and opening we are doing over here and will be lifting over here. So this is some sort of your unrealistic behavior. We know that eventually realizes mistake and you will be set its button and eventually it will come to this particular state. One it is coming to this particular state, you see that eventually it will go to heat up the coil.

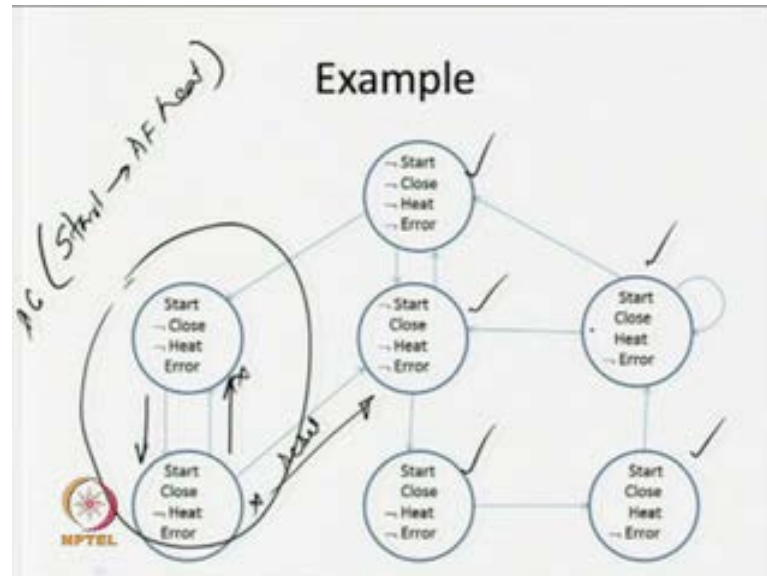
(Refer Slide Time: 18:48)



So this is I am talking about that it may have some contain unrealistic behavior. So, this is unrealistic behavior, but we cannot avoid it when modeling our system. So now what will see that? Where I can go a with this particular unrealistic behavior. So, that is why we are going to say that, we put some constraint while doing the model checking, so that it is going to remove those particular unrealistic behaviors. So that constraint that we are going to put is basically known as your pan is constant; that means we are going to look

for a fair system. We are going to put some fairness constraint in this particular with the help of this fairness constraint, what we are going to do?


(Refer Slide Time: 19:28)



(Refer Slide Time: 20:07)

### Model Checking with fairness

- It may sometimes be better to stick to the original model and to impose a filter on the model check.



We are going to eliminate such type of behavior that means, we are going to look the model only in fair path. So because eventually you know that, user will realize the mistake and it will press the reset button and it will come to this particular state. Once it coming to this particular state eventually coil will be heated up; so this the notion that we have talking about the fairness constraint. So will put some constraint which will filter on



the unrealistic behavior of the system and it is going to do the model checking on the fair percentage. That is why we are saying it is a model checking with fairness constraint. Now how we are going to do these things? So we are going to filter out those particular unrealistic behaviors.

(Refer Slide Time: 20:12)

**Model Checking with fairness**

- We verify  $M, s \models \psi \rightarrow \phi$ , where  $\psi$  encodes the refinement of our model expressed as a specification.

Handwritten notes on the slide:

- $M, s \models \phi$
- $M, s \models \psi \rightarrow \phi$  (with  $\psi$  underlined and an arrow pointing to it from the label CTL below)

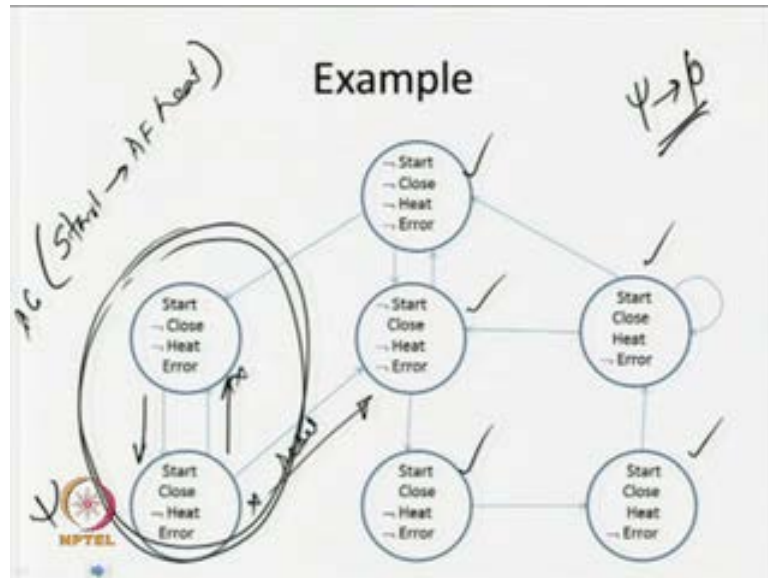
MPTel logo is visible in the bottom left corner of the slide.

So in this particular case, how to filter out this particular unrealistic behavior? We are going to again do it with the help of our CTL formula only. Now, what will happen? When I am going to model check a particular formula in a particular state  $s$  say  $M, s \models \psi$ . In this particular case, instead of modeling  $M, s \models \psi$ , we will try to verify  $M, s \models \psi \rightarrow \phi$ . So, what we are going to do? We will going to model check  $M, s \models \psi \rightarrow \phi$ . What does it means? Where  $\psi$  encodes the refinement of our model expressed as a specification.

So, what is this particular  $\psi$ ? It is nothing but a refinement of our model and we are expressing this particular refinement with the help of another specification, so which is another CTL formula. So, we are trying to capture this things that refinement of our model with the help of a CTL formula. Now, we see how we are going to do it? So in this particular case what will happen? Somehow we have to put some constraints, so that it is going to eliminate this particular unrealistic behavior. So we get trying to capture this particular unrealistic behavior with another CTL formula  $\psi$ ; that means we will say that when  $\psi$  is true then  $\phi$  is true, that means  $\psi$  implies  $\phi$ . We are going to check

for the correctness  $\psi$  implies  $\phi$ . That means, whenever our  $\psi$  is true then look for the correctness  $\phi$ .

(Refer Slide Time: 21:23)



(Refer Slide Time: 21:45)

### Model Checking with fairness

- We verify  $M, s \models \psi \rightarrow \phi$ , where  $\psi$  encodes the refinement of our model expressed as a specification.
- If  $\psi$  is true infinitely often, then  $\phi$  is also true infinitely often.

NPTEL


So in this particular case, what we can say that? If  $\psi$  is true infinitely often, then  $\phi$  is also true infinitely often. So, that means  $\phi$  is true infinitely often if it is infinitely often false then  $\phi$  cannot be true. But, if  $\psi$  is true infinitely often then we are going say that  $\phi$  is true infinitely often. So, we are capturing this behavior with the help of this particular formula model and in state  $s$   $\psi$  implies  $\phi$ , where  $\psi$  is the refinement of our

model expressing as a CTL formula and  $\phi$  is the formula or the property that we are going to check in our model. So we are saying this is the model checking with fairness and this  $\psi$  is going to basically give us the fairness constraint and if  $\psi$  is true infinitely often; we will say that  $\phi$  is also true infinitely often.  $\psi$  is not infinitely often true that means, it is thus going to through some unrealistic behavior. It will be in some looks, where this particular  $\psi$  is not true, but we know that in the system eventually it is going to come out this particular look. So, that is why we are trying to filter out those particular path and we are going to filter out with the help of this particular constraints  $\psi$ .

(Refer Slide Time: 23:01)

**Model Checking with fairness**

- Let  $C = \{\psi_1, \psi_2, \dots, \psi_n\}$  be a set of  $n$  fairness constraints.
- A computation path  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$  is fair with respect to these fairness constraints if for each  $i$  there are infinitely many  $j$  such that  $s_j \models \psi_i$ .
- We write  $A_C$  and  $E_C$  for the path quantifier  $A$  and  $E$  restricted to fair paths.



Now, in general now what we are going to say that? Instead of one particular fairness constraints we may have several fairness constants. So, we are going to say that let  $C$  is equal to  $\psi_1, \psi_2, \dots, \psi_n$  be a set of  $n$  fairness constraints. So, we may have several fairness constraint and you say that,  $C$  is set of several fairness constraints.

So, we will say that a computation path  $s_0, s_1, s_2$  like that is a fair with respect to this fairness constraint; if for each  $i$  there are infinitely many  $j$  such that  $\psi_j \models \psi_i$ . So, that means, we are going to look for a path in that particular path, those particular  $\psi_i$  will be true infinitely often. So in general situation what will happen?

We can say that and we write these things as  $A_C$  and  $E_C$  for the path quantifier  $A$  and  $E$  restricted to fair path. So,  $A_C$  basically says that this is all path through this particular fairness constant that means all fair path.  $E_C$  there exist a path with respect to given


fairness constraints  $C$ ; that means we are looking for a particular fair path. So, we are having a set of fairness constraints, now we are going to have the path quantifier  $A_c$  and  $E_c$  instead of general  $A$  and  $E$ .

(Refer Slide Time: 24:27)

### Model Checking with fairness

- We write  $A_c$  and  $E_c$  for the path quantifier  $A$  and  $E$  restricted to fair paths.
- $M, s_0 \models A_c G \phi$  iff  $\phi$  is true in every state along all fair paths.
- Similarly  $A_c F$ ,  $E_c U$ , etc.

$\underline{AG\phi}$       $AG\phi$



Now, we write  $A_c$  and  $E_c$  for that path quantifier  $A$  and  $E$  restricted to the fair paths. Now, in this particular case now what will happen? We can say that we are having the formula  $AG\phi$ ; so that means whether in all paths globally  $\phi$  holds or not. So this particular formula if we will write,  $A_c G \phi$ . What does it mean? Whether this  $\phi$  holds globally in all fair paths; that means, in all parts where this particular fairness constraints are satisfied.

So, that means we are going to look for all fair paths. That means the fairness constraints we may have such type of scenario of  $M, s_0 \models A_c G \phi$ , if  $\phi$  is true in every state along all fair paths. So similarly, we can define for  $A_c F$ ,  $E_c U$  etcetera; that means;  $AG\phi$  is all paths globally  $\phi$  holds without any fair constraint, but  $A_c G \phi$  says that  $\phi$  holds globally in all fair paths. That means, in all the paths where the fairness constraints  $C$  is true. So, this is the way that we are going to look in to it.

(Refer Slide Time: 25:44)

### Model Checking with fairness

- A computation path is fair iff any suffix of it is fair.

The diagram shows a sequence of states  $s_1, s_2, s_3, s_4, s_5$  connected by arrows. A path  $\pi$  is indicated by a diagonal line. A suffix  $\pi_3$  is highlighted with a bracket, and the word "fair" is written next to it. Other handwritten notes include  $\pi_3$  and  $\pi_2$ .

Now we can say that a computation path is fair if any suffix of it is fair. So, we can say that now you say what happened? See I am going to say that this is a path say I am coming from  $s_1, s_2, s_3, s_4, s_5$  something like that. I can say that this path is fair enough if any one of suffix is fair; that means if I can establish that. This is the suffix of the given path say this is your path  $\pi$ , then  $\pi_3$  is the suffix of this particular path starting at this particular state  $s_3$ .

(Refer Slide Time: 26:58)

### Model Checking with fairness

- A computation path is fair iff any suffix of it is fair.
- $E_c[\phi U \psi] \equiv E[\phi U (\psi \wedge E_c G T)]$
- $E_c X \phi \equiv EX(\phi \wedge E_c G T)$

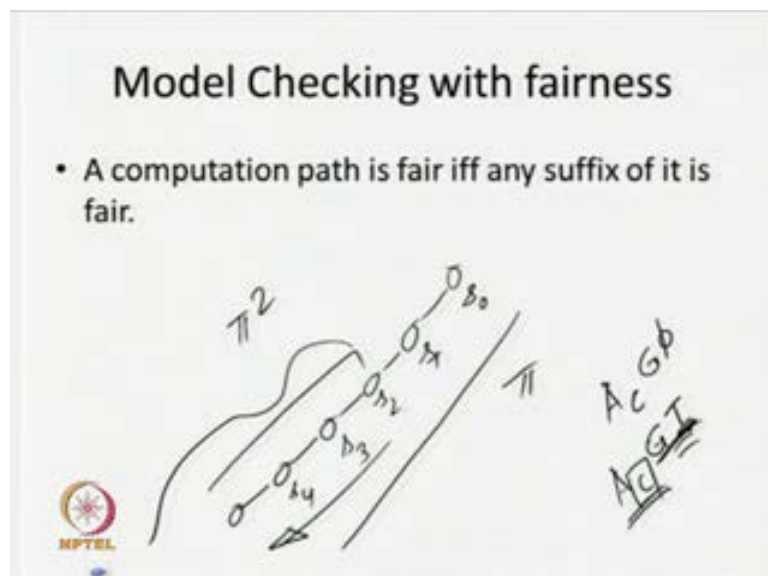
The NPTEL logo is located at the bottom left of the slide.

Now if  $\phi_3$  is fair then we can say that  $\phi_2$  is also be fair, because since it suffix is fair. So we are going say that that computational path is fair if any suffix of it is fair. If, we can establish the any suffix is fair then, we can loop into that particular path will also be fair; because ultimately eventually it will go to that particular of the suffix. So, in this particular case now, what we can say that? We can look for your model checking formula similar to something like that.

(Refer Slide Time: 27:08)



(Refer Slide Time: 27:24)



So, if I am going to say that any suffix of a path if it is your fair suffix; that means we can say what I say in this particular case? To what we are saying that the computational path is fair if any suffix of it is fair.

So, basically if you consider any path something likes that,  $s_0, s_1, s_2, s_3, s_4$  like that. So if this is path  $\phi$  and if you consider any of its suffix; say this is your  $\phi_2$ . If, you say that  $\phi_2$  is fair, then we can eventually say that  $\phi_1$  will be fair and we can say that  $\phi_0$  or  $\phi$  will be fair. Because eventually it is going through this particular pair path. So, that is why we can say that if in our formula, what we are looking into it?  $A c G T$ . So, we are saying that  $A c g \phi$ , now if I say that  $A c G T$ .

So, what does it means? That means in all state; we know that in all sates it is labeled with your truth value true. So that means if you say that  $G T$  basically talk about a path fair, true is always true. But, along with that instead of  $A$  we are saying that  $A c$ , that means it is the path where this particular fairness constraints is true. That means eventually, we can write that  $A c G T$ ; so, this is basically talk about this particular suffix, where the suffix abbreviate true is true. But, it is your  $A c$  means that means the fairness constraints  $C$  is also true over here. That means all these things we can say that if a suffix is fair then, this path will also be fair.

(Refer Slide Time: 29:11)

**Model Checking with fairness**

- A computation path is fair iff any suffix of it is fair.
- $E_c[\phi U \psi] \equiv E[\phi U (\psi \wedge E_c G T)]$
- $E_c X \phi \equiv EX(\phi \wedge E_c G T)$

Handwritten annotations on the slide include:  $A c G \phi$  (underlined),  $E_c G T$  (circled), and  $E_c G \phi$  (circled).

So, in that particular fairs what we can say now? Say we are going to look for say  $E c i$  until  $\psi$ . So, this is basically there are exist a path where this particular fairness

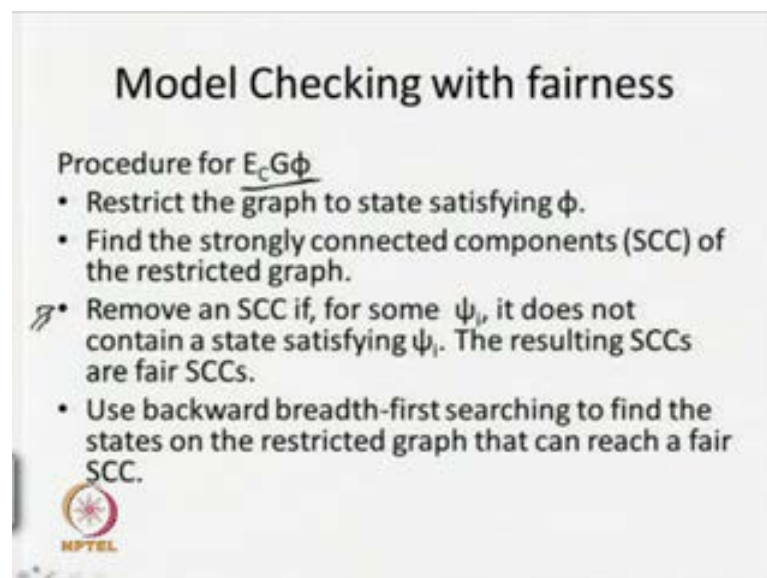


constraint is true,  $\phi$  until  $\psi$ . That means we are going to look for this  $\phi$  until  $\psi$  in fair path. So, we can say that this is equivalent to there are exist a path  $\phi$  until  $\psi$  and  $E c G T$ .

So, this is your fair path when true is always true. That means we can say that eventually when I am talking about that  $E c$  that means, there are exist a path with fairness constraint  $\phi$  until  $\psi$ . We are talking use in the general  $E$ , we that general  $E$  until operator without any constraints. So, we are going to look for  $\phi$  until  $\psi$ , but along with this particular fair path  $E c G T$ . So similarly for  $E c X \phi$  what we can write, that  $E X \phi$  and  $E c G T$ . That means, we are going to look for a next state, when it is the fair enough.

So,  $E c G T$  basically talk about the fair path; that means the path where this particular fairness constraints  $C$  are true. So you just see that it is your  $E c$  until operator and  $E c X$ , that means until there exist a path with the fairness constraints until and there exist a path with fairness constraint next state. These two are can be represent that with the help of  $E c G T$ , that means eventually will find that we need a procedure for either  $E G T$ ,  $E c G T$  or  $A c G T$ .


(Refer Slide Time: 31:52)



**Model Checking with fairness**

Procedure for  $E_c G \phi$

- Restrict the graph to state satisfying  $\phi$ .
- Find the strongly connected components (SCC) of the restricted graph.
- ✂ • Remove an SCC if, for some  $\psi_i$ , it does not contain a state satisfying  $\psi_i$ . The resulting SCCs are fair SCCs.
- Use backward breadth-first searching to find the states on the restricted graph that can reach a fair SCC.



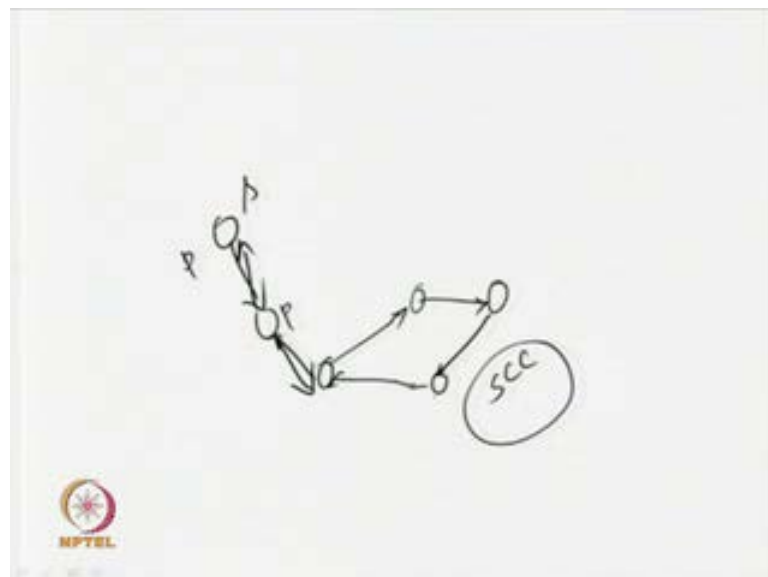
So basically what happens? We need the procedural for either  $E c G \phi$  or  $A c G \phi$ . That means once we have a procedure for  $E c G \phi$ , then we can talk over  $E c G T$  or  $A c G \phi$  we can talk about  $A c G T$ . So we need procedural for this one and even after



what happen?  $E \text{ until } E X$  can be express with the help of  $E \text{ c } G T$ . So we need basic procedural for whether  $A \text{ c } G \text{ phi}$  or  $E \text{ c } G \text{ phi}$ . If we have the procedural then we can have the model checking procedure for with fairness constraints.

Now what is the procedure for  $E G \text{ phi}$ ? Already we have select it we have another procedure. We are saying that restrict the graph to start, to state satisfying  $\text{psi}$ . So first step is we are restricted graph to a state where it satisfy  $\text{phi}$ . Then find the strongly connected components SCC of the restricted graph. And as a first step we are saying that use backward breadth-first searching to find the states on the restricted graph that can reach a S C C. So, with the help of this procedure we can take about  $E G \text{ phi}$ .

(Refer Slide Time: 32:28)



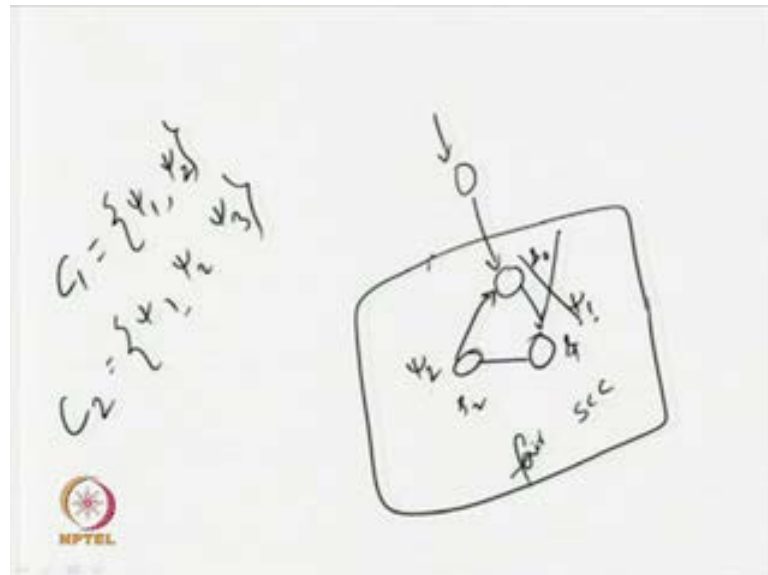
So, basically what happens? We can say that if I am having some states. So, you can say that this is your S C C, because all state can rich form any other state. Now form this particular state what will happen? You will follow the backward low, sorry my role in the direction. Because breadth-first search I can get any state, then we can say that this is basically you can reach over here; because if we are looking for  $p$ .  $P$  is true over here, because, it will be  $p$  is true will also be over here. So in this particular way, we are going to collect all the states.

Now in faired what will happen? We are going to look for  $E \text{ c } G \text{ phi}$  there exist a path with fairness Constraints  $G \text{ phi}$  is true. Now, this is also similar to that procedure only,

what we are going to say? The first step, restrict the graph to state satisfying phi find the strongly connected component SCC is of the restricted graph, already we have discussed.

Now we are having one particular condition over here. Remove an SCC if, for some psi i it does not contain a state satisfying psi i. Now the resulting SCCs are fair S C Cs. Now will just give an example; so, we are saying that infinitely open all psi as must be true. So, if it is not true, then we are going to remove does particular as S C Cs. Again the fourth step you similar use backward breadth-first searching to find the states on the restricted graph that can reach a fair as S C C.

(Refer Slide Time: 34:17)



Now, you just see that. Now I am giving an example say; so I now that these are the state s 0, s 1 and s 2. So this is an S C C, we do not have any problem. Now you just see that I am having a fairness constraints C is equal to say psi 1 and psi 2. Now want to happen say, in this particular step s 1 say psi 1 is true and in this particular state say psi 2 is true. Now, in this particular case, whenever I am coming to say from several, states you coming to this particular state. Then what you will happen? You just see that it will be in this particular loop it can go a verse, so infinitely it can go over hear but, wherever it go some point of times psi 1 will be true or psi 2 will be true. So, I can say that this is an fairness is say. But if I talk about fairness constraints say this is C 1 and C 2 and I am taking say that psi 1, psi 2 and psi 3 are fairness constraints.


Now, where I come to this particular S C C? In this S C C, wherever I go I will find that  $\psi_1$  will be true; whatever I go, I will find that  $\psi_2$  will be true. So, in none of the state in this particular S C C, strongly connects the components  $\psi_3$  is true. So, let me if I remind in this particular loop then  $\psi_3$  is not going to with true. So, that means this fairness constraints it is not satisfied over here, so I am not going to say that this a fair S C C.

(Refer Slide Time: 31:55)

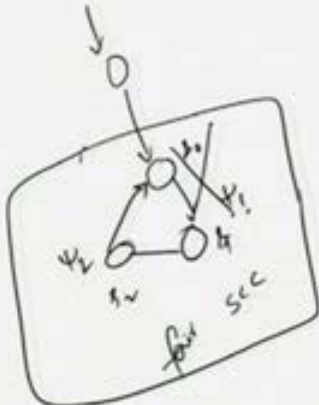

### Model Checking with fairness

Procedure for  $E_c G \phi$

- Restrict the graph to state satisfying  $\phi$ .
- Find the strongly connected components (SCC) of the restricted graph.
- Remove an SCC if, for some  $\psi_i$ , it does not contain a state satisfying  $\psi_i$ . The resulting SCCs are fair SCCs.
- Use backward breadth-first searching to find the states on the restricted graph that can reach a fair SCC.



(Refer Slide Time: 34:17)

So as per third step of my algorithm we are going to remove this particular SCC from our graph; so this is the thing that we are talking about that remove. And SCC if some  $\psi_i$  it does not contain a state satisfying  $\psi_i$ . The resulting SCCs are the fair of S C C.

So, that means the fair S keeping only the fair S C C. So, if this fair S C C, that means whatever fairness constraints we are giving close will be true over here. So this is the models checking algorithm with fairness constraints. You just see that, if once we have this procedures A G or E G then other operators can be define with the help of that A G or E G.

So, eventually we have seen that, we have got a procedure form model checking with fairness and we have discuss hear with  $E_c G \phi$ . So, this is similar to your E G  $\phi$  except that we are having one particular step over here.

(Refer Slide Time: 37:12)

**Model Checking with fairness**

- A computation path is fair iff any suffix of it is fair.
- $E_c[\phi U \psi] \equiv E[\phi U (\psi \wedge \underline{\underline{E_c G T}})]$
- $E_c X \phi \equiv \underline{\underline{EX(\phi \wedge E_c G T)}}$

Handwritten notes on the right side of the slide:

- EU
- EX
- AG/AF
- AG
- EG

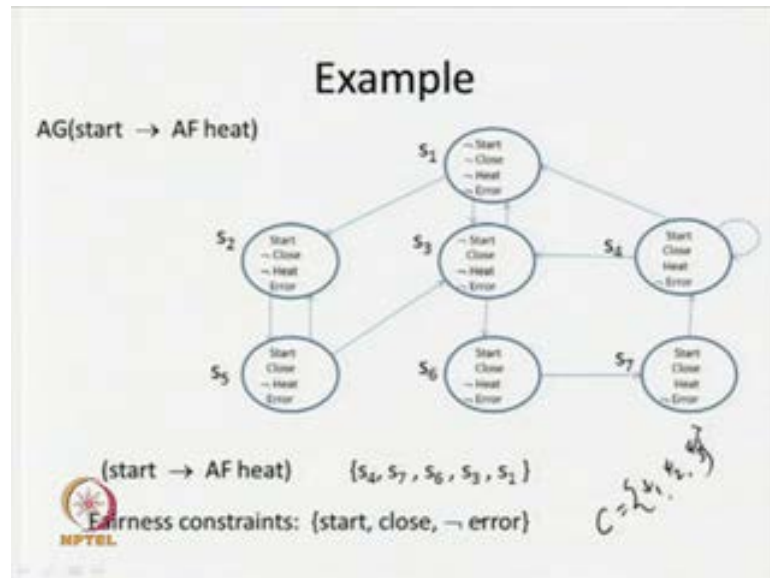
NPTEL logo is visible in the bottom left corner.

Where we are going to remove the unfair SCCs or we are going to keep the fair S C C. Now you just see that what are the procedure that will be needing for your model checking algorithm with fairness. Say we need either one E until or E x or say A G or say A F.

Now, we have seen that with only A c G or E c G. If we now the procedure for any A c G and E c G, then we can look for a procedure for E, when E 1 E X because, E c is again express with the help of that normal E operator and E c X again express with the normal

EX operator. But, along with that we are having in this particular fair suffix. So, EC GT and EC GT, that means whether fairness constraint of fair. Basically we are going to keep all that fair SCCs, so this is the notion; so we are having. Now we have seen procedure for EC G or similarly we can construct for AC G also.

(Refer Slide Time: 38:10)

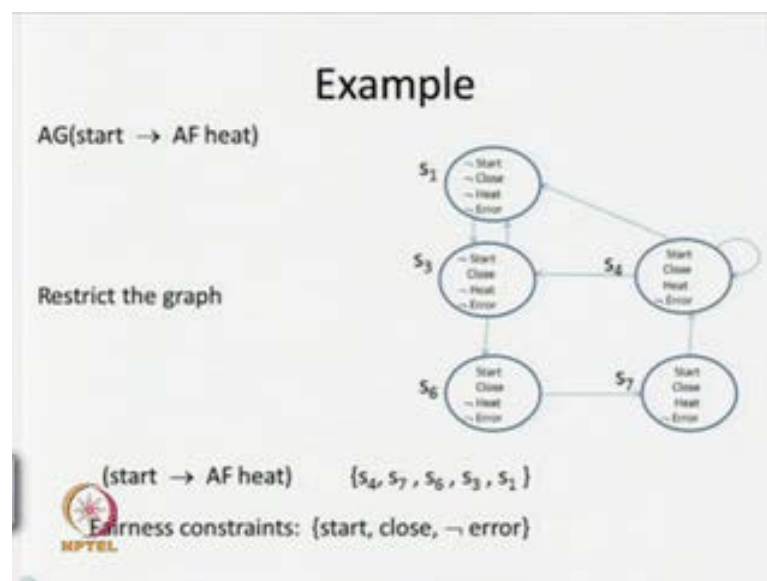


Now, after knowing these particular fairness constraints let us come back to our example, that we are talking about that microwave oven examples. And we have come up with this particular model, which is having seven states and we try to state for this particular formula  $AG \text{ start} \implies AF \text{ heat}$ . Now what we apply our models model checking algorithm, what we found that? These formulas just not satisfying this particular model. Now one option is it should go for defining of this particular model, come up with a (( )) solution or second we have seen that, we can go array or remove or prevent those particular unrealistic behavior; and try to check this particular formula in realistic behavior only. And for that we have to apply fairness constraints. Now we have seen that this is some sort of unrealistic behavior, that closing and opening the door repeatedly after starting it. So we are trying to remove this particular unrealistic behavior.

So, how you are going do this thing? That means we are going to do it would have of some fairness constraint. Now what will be the fairness constraint over here? So, what is the fairness constant? That we are going to look for it start, close and not of error; that means, when it go to start and close situation, it should not go into the error condition. So

not of errors, so this is basically that we have talking about the C is the set of fairness constant, where I am having psi 1, psi 2, psi 3 as my fairness constraint. So here the fairness constraints are start, close and not of error and we are going to look for A G start implies A F heat. Already we have seen the marking of start implies A F heat and we know that these are the states where start implies A F heat is true; s 4, s 7, s 6, s 3 and s 1. Now along with that we are giving this particular fairness constraints; start, close and not of error. Now we will see, how that model checking with fairness constraint will be use to check for this particular formula.

(Refer Slide Time: 40:22)



Now, what will happen in this particular case? Now as for these things, we are going to restrict the graph with the states where this particular formula, start implies A F heat is true. So basically we will find that, these are the states where this particular formula, A F start implies A F heat is true. Because in the order two states s 2 and s 5 this formula is not true.

Now after that what happens? We have to look for those particular fairness constraints. Now as per our example now we will see that; what are the SCCs over here. Once you will get the S C Cs, then we will see whether in this particular SCCs start, close and not of errors are true or not.

You just see that wherever it go. You can look for this particular scenario say I am having, this is a loop; so wherever you go, you can go to any other states. So in this

particular case, what will happen? You will find that start is coming true, close is coming true or not of error is coming true. So, infinitely open it is coming. On the other hand you can say that, I can be in this particular loop. So, one of this particular fairness constraint will get true.

So, like that we can say that this is the remaining SCCs that we are going to have; and if you now apply this particular  $A G \text{ start } A F \text{ heat}$ . Then will find that all those particular states, this particular formula is true. That means globally in all path gobbling start implies  $A F \text{ heat}$ . That means whenever you start the oven eventually it will heat up the coil. Because here, what happens basically? With the help of these fairness constraints, we are removing this particular unfair path.

So, I am going to check only all those particular fair path. Now you just see that with fairness constraints eventually you say that, ok our model is going to work and it is going to satisfy this particular formula. But, in some point of time may be user is going to be in this particular loop forever then this not but, we know that user also use his own intelligence and eventually he will press restart button and he will come to this particular point.

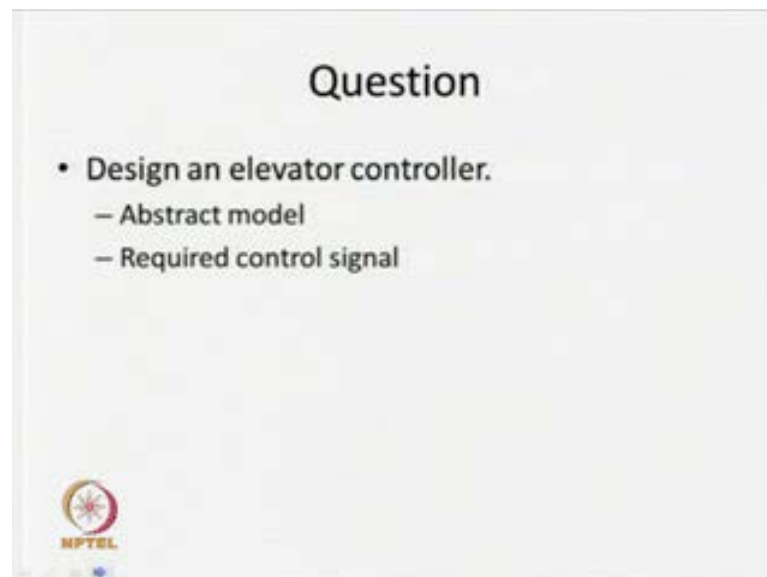
So, this is some sort of your unrealistic behavior. And it constraints fairness constraints, we are removing it and we are trying to check our property in the remaining graph. So, this is the way we can check for all correctness, one is your refine your model or secondly try to find out the constraint, fairness constraints and model check on the fair paths only. So, this is basically we have discussed about the model checking algorithm, we need one model. We need the properties; those properties will be express in the CTL. So these two will give as input to the model checker and the model checker will give the set of steps, where it is particular formula is true.

And sometimes what will happen? If we can identify that, it is having some unrealistic behavior but, it is unable able then we will try to identify some fairness constraint and we are going to look for the correctness of the property in fair paths only. Fair paths will be those path where those particular fairness constraints are true. And with this we can now proceed for a model check and we are going to check whether this property is true in these particular fair paths or not. So this is in general or we can say that this is the things

or this is the concept about the model checking and in any design we can apply this particular model checking algorithm before proceed for in our design cycle.

So, what is this? Some come up with your design, express your properties in CTL and try to check those properties in those models with the help of model checking algorithm. But, if the system is having some unrealistic behavior, then we try to filter out those particular unrealistic behaviors with fairness constraint and we will do the model checking on fair paths only. So, this is about the model checking approach for verification of our properties.

(Refer Slide Time: 44:30)



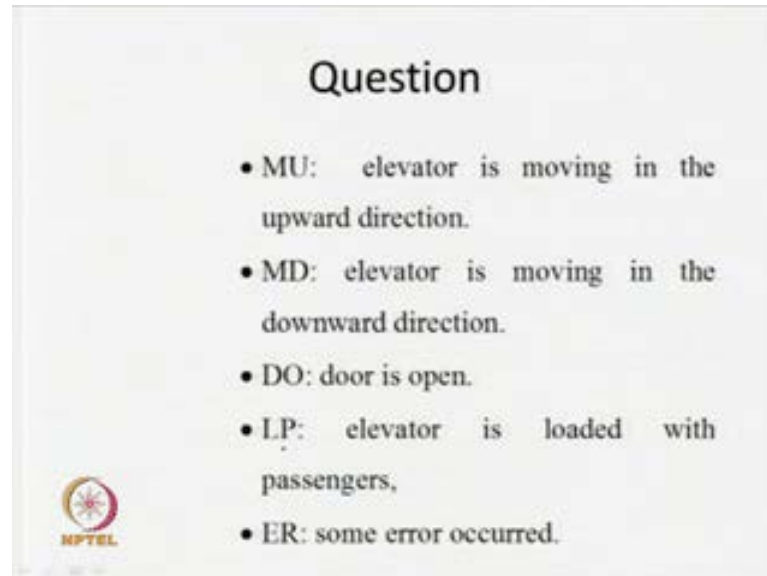
Now, as a now we are having now enough information about your model checking algorithm and we know how to do it; now just look for some questions. Now that one question that I am giving you know; design an elevator controller. I think in a multi stored building all of you have used lift or elevator and what we are having that, you can press button to go to any floor or it can give a request to the lift by pressing a button in that particular floor.

So, for that we have to design this controller. So, we have to identify, what are the things or what are the control signals required and depending on that we are going to design our controller. In this particular case again as per our design principle, we are going to abstract of the model and we are going to identify the required control signal; and with the help of those control signals, we are going to have a model. Once we have the model,




then we are going to look for the properties that need to be satisfied by our controller and we are going to check those particular property.

(Refer Slide Time: 45:38)



**Question**

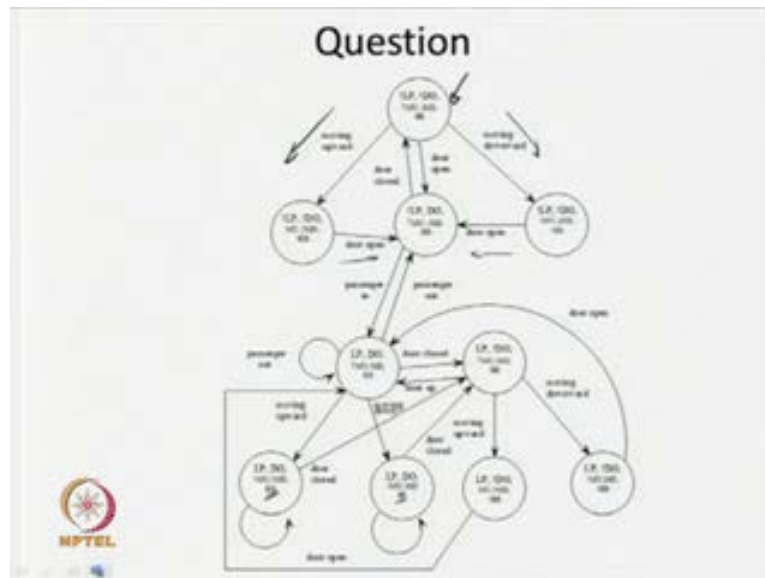
- MU: elevator is moving in the upward direction.
- MD: elevator is moving in the downward direction.
- DO: door is open.
- LP: elevator is loaded with passengers,
- ER: some error occurred.



In this particular case, I am coming with a very simple solution or a I am just abstracting out the controller. In such way that we are having a very less number of control signals, so this is the initial design. After that you can go for refine input more and more information into our model. So, in this particular case I am coming up with four signals; one I am talking about M U: that means elevator is moving in the upward direction. M D: this is another event, I am going to say this elevator is moving in the downward direction; so these are the signals indicating the lift is moving upward or lift is moving downward. D O: basically door is open. Door can be either in two conditions, it is open or close; so I can identify whether the door is open.

L P: elevator is loaded with passengers. And E R: it has gone to some error condition, some error is occurred. So, we have identified these particular five control signals, so these are the atomic proposition whether this will be true or false. Now, with that help of this thing, we can come up with a model.

(Refer Slide Time: 46:48)



So, this is simplified model, I am saying that this is one particular state. What we are saying that? This is knot of L P, knot of D O, knot of M U, knot of M B and knot of E R; that means, it is in the ideal scenario. Now depending on these things when the moving upward or moving downward, then what will happen? Some of the signals will go as a move upward will be true, this is moving downward and depending on these things door open and door close. These are the events and depending on that, it is going into define scenario and different states. And it is all the states are labeled with the control signal, which are true over there you can check it.

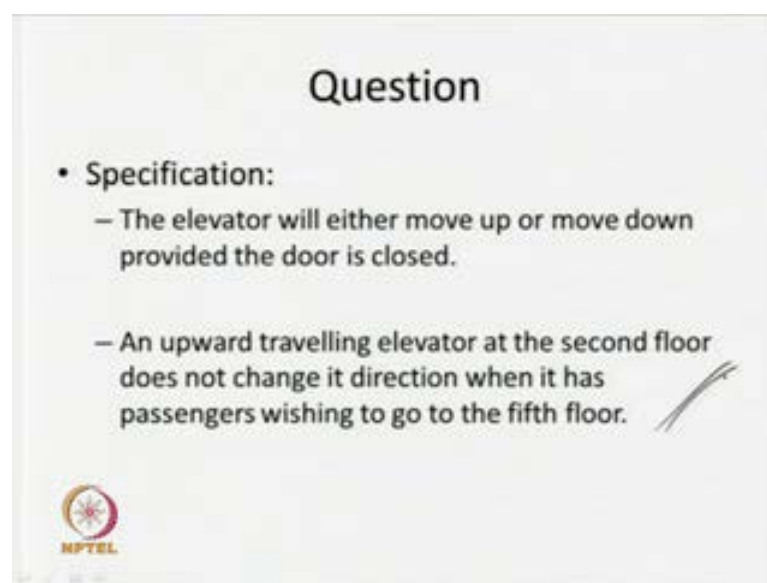
Similarly, presence are in presence are out, these are the events depending on that. It will go to some other state and after that we can close the door; one close the door, then it will move upward or move downward. And without closing the door if you again say trying to move up or move down, it will go to the error scenario, so these are error scenario by looking into the scenarios that and even that is going to up and in while operating this particular leave the elevator. We can come up with this particular simplified design. Once we have this particular model, now we can look for properties that need to be satisfied with this particular model and we can apply this particular model checking algorithm to check for the correctness of our design.

If, the design is correct then we can proceed further. For fabrication other things if it is not correct, then what will happen? Our model checking algorithm or model checking

algorithm give a counter example and if you say that, if you follow a particular path or particular execution test the properties are not true.


And as a designer what happens? We can revisit our design or we can rectify those particular designer or on the other hand. If as a designer if you combines that this particular behavior is an unrealistic one. Then I try to identify some fairness constraints and I will apply the model checking with those particular fairness constraints. So, this is the way we are going to proceed.

(Refer Slide Time: 48:57)



**Question**

- **Specification:**
  - The elevator will either move up or move down provided the door is closed.
  - An upward travelling elevator at the second floor does not change its direction when it has passengers wishing to go to the fifth floor.



Now, once we have the model then what we can say that? What may be the specification? What are the properties? So, one simple property I am saying that, the elevator will either move up or move down provided the door is closed. That means, the elevator should not have any movement if the door is opened. So, these properties must be satisfied by our controller.

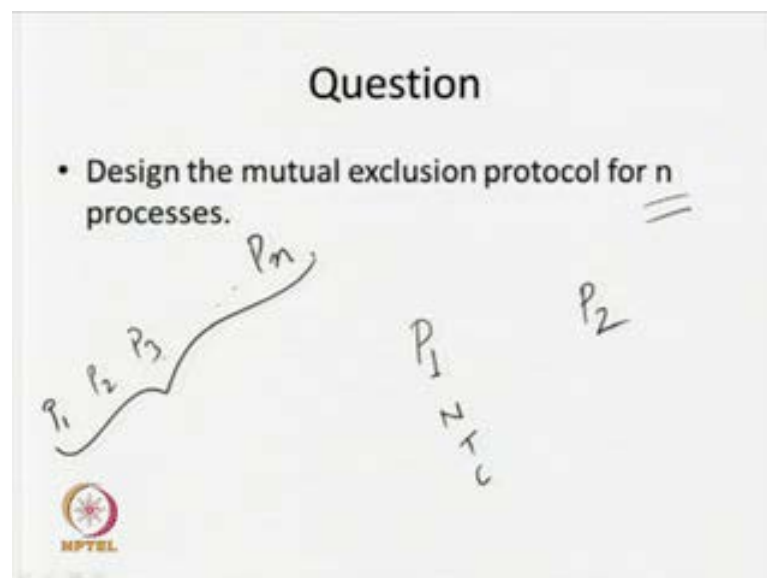
So, another property I can say that an elevator travelling, say an upward travelling elevator at the second floor does not change its direction; when it has passengers wishing to go to the fifth floor. So, this is basically the algorithm that you have incorporated in our lift controller. Because we should have, some protocol to give the service to the passengers. So generally what happens? The simple procedure, a simple algorithm or simple notion, I can say that if lift is moving upward genuine level generally it is going to service the request for all upwards direction.

Why? Because, as a designer we know that, if change lift is moving up and form some through, if we are going to come down; then what will happen? There will be change of directions in the motor, so changing the direction of motors is going to consume power, energy. So, in general notion what happens? We are going to keep the service to all upward direction first, if the lift is moving upwards; then we change the direction then the lift is coming in the downward direction. So, that is why such type of properties we need to satisfy to check for the correctness of protocols. But, if you are incrementing the some other protocol then this property may not be true.

So, depending upon the protocol that we have implemented or that if we are using in our lift controller. We have to identify the properties and we need to check those particular property. So, these are the specification.

Now, what happens, you just see that. You are having the model, you have come up with a model. We have those particular specifications, now we have to represent this particular specification in your temporal logic formula. Now, we have having enough knowledge about your temporal logic, about CTL. Now this is you take this as a tags and try to write down the CTL formula for this two particular specifications.

(Refer Slide Time: 51:19)



Now another question you just see that, what happens? Already we have discussed about mutual exclusion protocol. So, in this particular mutual exclusion protocol we have come with a model and what we have seen that? Or what we have discussed that? It is having

two processors  $p_1$  and  $p_2$ . So we are working with two processors  $p_1$  and  $p_2$  and what are state? We are saying that it may be in your none critical reason, it may be time to entire critical reason or it may be in the critical reason.

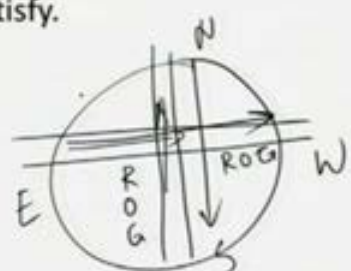
So, we have model with and we have seen, what are the properties we need to satisfy? Now, I am giving a question to you people, now is a check design the mutual exclusion protocol for  $n$  processors. We have the mutual exclusion protocol for two processors, now I am saying that you extend this model for  $n$  processors.

Now, what will happen? Now we are having two processors;  $p_1$ ,  $p_2$  and we know the states. And eventually what we are doing? We say that come up with a combined model and you do your job. Now, instead of two processors, now I am having  $n$  processors;  $p_1$ ,  $p_2$ ,  $p_3$  like that  $p_n$ . Now come up with a model and see how to model this one and how to model check those particular property.

(Refer Slide Time: 52:38)

**Question**

- Design a controller for Traffic light.
- Mention the property that the traffic light controller should satisfy.



NPTEL

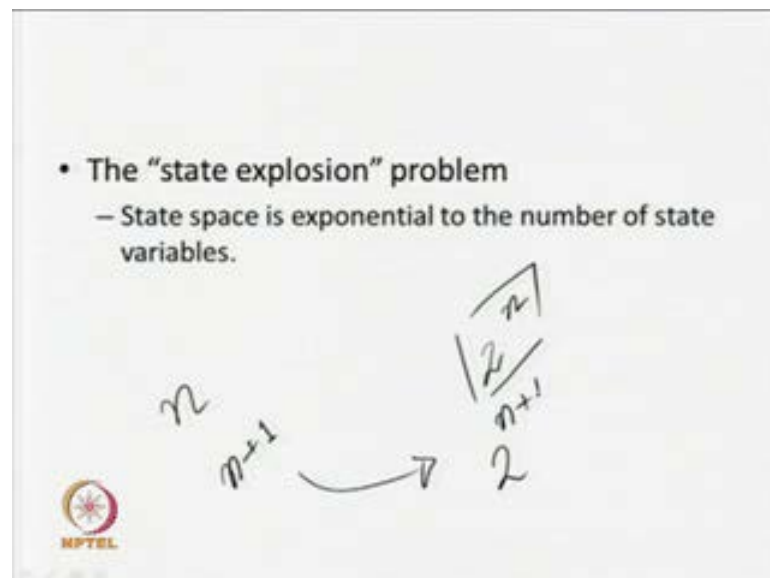
So, similarly I can give some more questions. You try to come up with a model and try to look for the specification or the properties that need to be satisfied. So, one another simple question. I can say that design a controller of for traffic light. So, we having a traffic light controller, now you should try to look or try to design this particular controller. So, that we can control the traffic in a junction and now try to come up with a specification of this particular traffic light controller. And represent those particular

specification in your CTL and after that apply CTL model checking algorithm to check those particular profile.

Now you just see that in just giving simple hints. So, in a traffic light controller what will happen? Say I am having a junction, road junction. So, it can say that it can go east-west direction or it is a north-south direction; now what will happen? Here I can have three signals say; red, orange and green. So, this is for east-west direction and similarly I can have red, orange, green spore east-west direction and say north-south direction.

Now, what will happen? Now we see that, what is the property? That both the red light should not glow simultaneously; on the other hand I can say that both the green light should not glow simultaneously or at same time. Because, if you are glowing the green light together, then what will happen? Car will move from both the right side and there may be collusion. So what will happen? If one or specification should be like that, both green lights should not be glow simultaneously. Similarly one is red if say north-south direction is your red then east-west direction will may go into green.

(Refer Slide Time: 55:03)



So, these are the specifications. Now we have to identify those specifications and try to write those particular specifications in you CTL and design the controller; so that we can control the traffic correctly in this particular junction. Now what we have seen? That we have coming up with this model checking method, model checking algorithm; we are

having properties, we are having a model and we are going to apply check whether those properties are true in the model or not.

Now how I am going to get the model? We may have a problem because, you just see that; if we are having  $n$  control signals, just say that if we are having  $n$  control signals. Then what will happen? How many define combinations we may have?  $2$  to the power  $n$  define combination; that means, we may have  $2$  to the power  $n$ , different possible steps. All may not be resemble, because we have already seen our mutual excursion protocol; where we all having  $6$  control signals, so total state space is two to the power  $6$ . But, all may not be resemble but we have to ready for a port situation. So I am having on control signal,  $n$  control signal then we are going for  $2$  to the power  $n$  different states.

Now say in our designing, if you have increase this particular control signal,  $n$  control signal power  $n$  minus  $1$ . Say we need is one more extra or additional control signal, then the number of state space will go to  $2$  to the power  $n$  plus  $1$ . So, if  $5$  control signals we are having  $2$  to the power  $5$ , which six control signals will be having  $2$  to the power  $6$ . So if it is  $32$ , next it will be your  $64$ . If we add one control signal it will go for  $2$  to the power  $7$ ,  $128$ . So this is the number of states depending on the number of control signals, and it is exponential in lesser. So, basically they are having a problem in this particular model checking algorithm. Because, the problem is known as a state exclusion problem; because, state space is exponential to the number of state variables.

Now though we have seen that, we are having an automatic procedure or automatic algorithm to go for model checking for CTL formula. But, another problem that we are having with this model checking is yours state exclusion problem. So now, there are some research works are going on how to contain or how to control this particular state space exclusion problem. So, in our next class or next module what we are going to see? We will see one particular approach to contain this particular state space exclusion. I will stop here today.