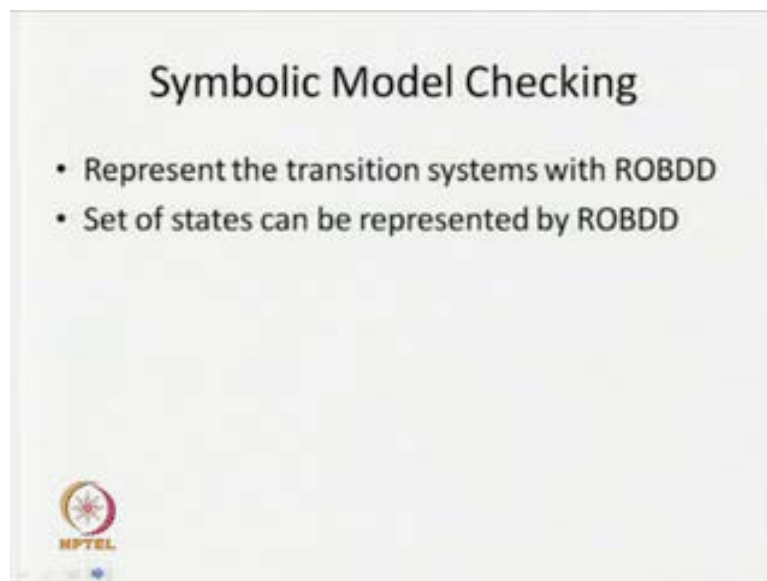


Design Verification and Test of Digital VLSI Designs
Prof. Dr. Santosh Biswas
Prof. Dr. Jatindra Kumar Deka
Indian Institute of Technology, Guwahati

Module - 6
Binary Decision Diagram
Lecture - 5
Symbolic Model Checking

So what we are discussing till now, we are discussing about a data structure call binary decision diagram or in particular we are looking into ROBDD reduced ordered binary decision diagram, and we have seen that with the help of these data structure; we can represent most of the all Boolean functions and it always gives a compact representation of our Boolean function.

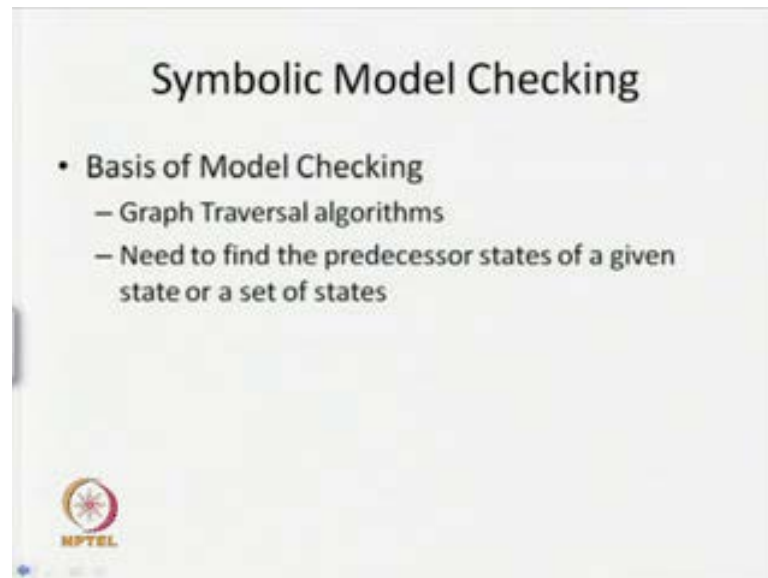
(Refer Slide Time: 00:46)



Also in last class, we have introduced, how we are going to represent transition system with the help of ROBDD, also we have seen that if we are going to take some set of states. We are having the entire transition system, we are looking for a particular set of states; then these particular set of state can also be represented by ROBDDs. So, we can represent the entire states phase, entire state transition diagram with the ROBDDs and set of states also can be represented with the help ROBDD and I have slightly introduce that, these particular data structure can be used to implement a model checking algorithm. And when we use ROBDDs or in particular OBDDs to represent our transition system,

when we are going to implement a model checking algorithm; then we call this particular model checking as your symbolic model checking, because we are symbolically representing the entire system.

(Refer Slide Time: 01:47)



So basically what happens in our model checking algorithm, what is the basis of model checking algorithm? If you look that, you will find that it is nothing but some graph traversal algorithm, because the Kripke structure they are using to model our system can be treated as graph. And in that particular graph, we are using some graph traversal algorithm to look for a particular state of set of states, where a given property is true, and the model checking algorithm written this particular set of states. So in this particular graph traversal algorithm, if you look minutely you will find that our basic requirement is to find a predecessor state of a given set of states or a given state. So this is the basis thing that we need to find out the predecessor states of a given state or the set of states.


(Refer Slide Time 02:40)

Symbolic Model Checking

- Important relationship between $\text{Pre}_{\exists}(X)$ and $\text{Pre}_{\forall}(X)$:

$$\text{Pre}_{\forall}(X) = S - \text{Pre}_{\exists}(S - X)$$

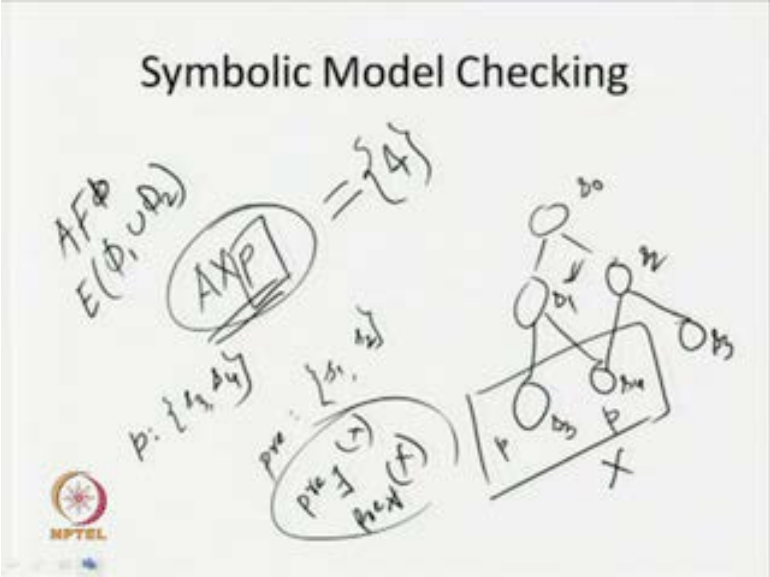
S: Set of all states
X: Subset of S



So for that what happens, we have to somehow find out the predecessor state.

(Refer Slide Time: 02:46)

Symbolic Model Checking



Say just like that, if I am going to consider this particular system, say this is your $s_0, s_1, s_2, s_3, s_4, s_5$ and say that the atomic proposition p is true in s_3 and s_4 . Now if you look into this thing, what you call? Say if we are going to find the states where AXp is true that means; in all states, in all path where in the next state p is true, so this is Xp in all path, next state p is true. Now in this particular case, first we are going to see the states where p is true. So in this particular case, we will find that, the atomic proposition

or a CTL formula p is true in state your s_3, s_4 that means; we are going to get this particular two state s_3 and s_4 and you can say that, this is a subset x .

Now in our model checking algorithm what will happen? We are going to find out the predecessor state of these two a states or say this is subset x . Then we are going to get the predecessor as your s_1 and s_2 ; s_1 and s_2 are the predecessors states of this particular subset x . Now for here what will happen? We are going to look for pre they are exist X and pre for all X , in last class I have introduce that means; for all x means all the transition is coming to this particular subset and pre they are exist means at least one x is coming to this particular subset x .

Now in this particular $A X p$, in all path in next state p is true that means; all the transition must come into this particular subset x . So in this particular case, now the predecessor state we are getting s_1 and s_2 . Now we are going to look for this particular criteria, whether all transitions are coming to this particular subset or not; we will find that for s_1 all the transition are coming to this particular state subset x but from s_2 one of the transition is not coming into this particular subset x ; it going outside of this particular. So that means, pre for all X will give me this particular state s_1 that means; I can say that $A X p$ in all path in next state p is true is in state s_1 . So we are going to get this particular state s_1 and in this particular state s_1 p is true. You just see that, in our model checking algorithm our basic requirement is to find the predecessor state.

If similarly, if I am going to look for that $A F \phi$ or say $E \phi_1$ until ϕ_2 , in all those cases what will happen? We are going to start from particular set of states and we are going to traverse this particular graph in backward direction, backward traverse basically, and we are going to collect more and more state; then we are going to say, we are going to find that which are a state that $A F p$ or $A F \phi$ or $E \phi_1$ until ϕ_2 we have prove. And eventually, we are going to get a set of states and our model checking algorithm is going to written this particular set of states and we have seen that, we can use ROBDDs to represent this particular set of states.

And ultimately, our model checking algorithm or symbolic model checking algorithm is going to return me those particular set of state in symbolically, which repre which will be represented with the help of our ROBDDs. So this is the basic thing, so when we are going to use ROBDD for model checking algorithm is to get the symbolic model checker

for basically, we need we need these two function; pre they are exist X and pre for all X, in last class we have introduce these two notion then I have explained it.

(Refer Slide Time: 06:50)

Symbolic Model Checking

- Important relationship between $\text{Pre}_{\exists}(X)$ and $\text{Pre}_{\forall}(X)$:

$$\text{Pre}_{\forall}(X) = S - \text{Pre}_{\exists}(S - X)$$

S: Set of all states
X: Subset of S

Handwritten Venn diagram showing a large rectangle labeled Pre_{\forall}(X) containing a smaller rectangle labeled Pre_{\exists}(X). The area between them is shaded, representing S - Pre_{\exists}(S - X).

Now also we have seen one important relationship between pre they are exist X and pre for all X and you can see that, pre for all X can be express by this particular expression s minus pre they are exist s minus x. So this is a relationship between our pre they are exist X and pre for all X, where s what is s? s is nothing but set of all state that means; we are going to consider all the states of my transition system and x is a subset of this particular s that means, we are going to look for what are the predecessor state from this particular subset x. And if we can calculate now pre they are exist x, then from if I know this particular things, then very well always I can calculate, pre for all X also while using this particular relationship that means; we need one predecessor to calculate pre they are exist x.

If we have this particular procedure to evaluate this particular set of states, then very well we can evaluate this one also by using this particular relationship. Now we are going to see, how we are going to evaluate this particular set of state that means; pre they are exist x, so we need a method to calculate this one and this is the basis of our symbolic model checking algorithm. If I give you a set of state, always my requirement is to find out order a predecessor state and out of that I am going to see, what are the states? Where it is going to satisfy? Pre they are exist or it is going to satisfy pre for all.

(Refer Slide Time: 08:37)

Symbolic Model Checking

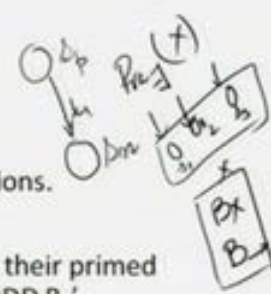
Procedure for $\text{Pre}_{\exists}(X)$

Given,


- B_x : OBDD for set of states X .
- B_{\rightarrow} : OBDD for transition relations.

Procedure,

- 1 - Rename the variables in B_x to their primed versions; call the resulting OBDD B_x' .
- 2 - Compute the OBDD for $\text{exists}(x', \text{apply}(\bullet, B_{\rightarrow}, B_x'))$ using the **apply** and **exists** algorithms.



$\hat{x} = \langle x_1, x_2, \dots, x_n \rangle$ $\hat{x}' = \langle x_1', x_2', \dots, x_n' \rangle$



Now in this particular case, now how we are going to look into it. So we are going to look for a procedure for pre they are exist X where what is the input? The input given to us is your B_x it is the OBDDs for the set of state. So we are having a set of states, say this is a set of state x , we are having some states say s_1, s_2, s_3 say something like that. So B_x we are going to say that, this is the BDD representation of this particular subset x and secondly we are going to say that, B_{\rightarrow} basically we are having that transition system and that transition system already we have seen or we have discuss how to represent a transition system with OBDDs or may be ROBDDs; so we say that B_{\rightarrow} is the OBDD representation of our transition system. So basically, we are getting two OBDDs; one is your B_x , this is the OBDD representation of our set of state x and B_{\rightarrow} is the OBDD representation of our transition system.

Now one basic requirement is there, for this B_x and B_{\rightarrow} what is the basic requirement? Already I have mention that, since we are going to use some operation on these two particular BDDs. So they must have computable variable ordering, already we have mention what is compatible variable ordering? They must have variable ordering. So we are taking two BDDs or OBDDs as our input and both the OBDDs must have computable variable ordering. Now what is the procedure? What we are going to do? Say this is very simple one I am going to explain it as for procedure we are going to say that rename the variables in B_x to their primes versions and we call this resulting OBDDs as B_x' . What is primed version of a variable because already we have seen that if we

are going to represent a transition system we need to set of variables to represent those particular transition.

So if I am having a transition from say s_p to say s_n , s_p is the my present state or current state and s_n is the next state of this particular transition say t_1 . So this transition will be represented by the order pair of 2 states, so basically this state, current state will be represent by some state variable say, if we need n state variable; then we are going to take the help of n state variable say from x_1 to x_n .

To represent this particular s_n , we need another set of variables and this set of variable basically, the primed versions of this particular original variables. Now what we are going to do? We have going to rename those variables in B_x to their primed version, I will explain it what we are going to get. Then we are going to compute an OBDDs for this particular operation, first we will apply, we will use this particular apply algorithm with this particular dot operator with this two BDDs, OBDDs for transition system and OBDDs for the set of states where the variables are rename to their primed version and this particular apply algorithm will give me an OBDDs and where the variable ordering is same with your B arrow and B_x prime.

Now after that what we are going to do? We are going to apply this particular exists algorithm where we are going to make it independent of this particular primed version of those particular variable. So x but basically it is a factor, I can say that what is your x set? x set is nothing but the factor of all variables that we are using x_1, x_2 to x_n . Similarly, x prime set is the factor of variable of prime version of this one x_1 prime x_2 prime like that x_n prime. So these are the two set, that you have to apply one renaming of the variable and second we are going to use this particular method first; using they apply algorithm with dot operator, then we are going to use the exists algorithm. So these are the two set, that we have to perform when we are going to get this particular p they are exist X .

(Refer Slide Time: 13:20)

Symbolic Model Checking

– Rename the variables in B_x to their primed versions; call the resulting OBDD B_x' .

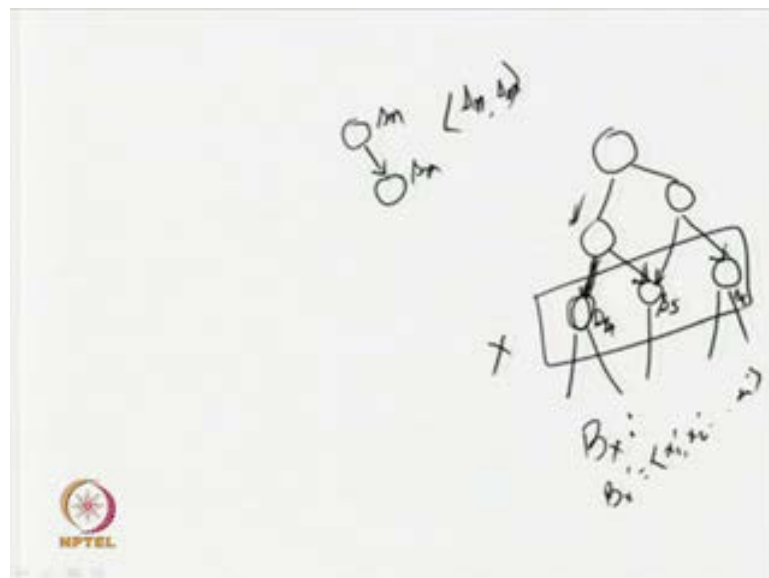
The image contains handwritten mathematical expressions and diagrams. On the left, a function is defined as $f = ab + a'c$. Below it is a truth table with columns for variables a , b , and c , and a row for the function value f . The values in the table are: $(a,b,c) = (0,0,0) \rightarrow f=0$, $(0,0,1) \rightarrow f=0$, $(0,1,0) \rightarrow f=0$, $(0,1,1) \rightarrow f=1$, $(1,0,0) \rightarrow f=0$, $(1,0,1) \rightarrow f=1$, $(1,1,0) \rightarrow f=1$, and $(1,1,1) \rightarrow f=1$. On the right, another function is defined as $f_2 = xy + x'z$. Below it is a truth table with columns for variables x , y , and z , and a row for the function value f_2 . The values in the table are: $(x,y,z) = (0,0,0) \rightarrow f_2=0$, $(0,0,1) \rightarrow f_2=1$, $(0,1,0) \rightarrow f_2=0$, $(0,1,1) \rightarrow f_2=1$, $(1,0,0) \rightarrow f_2=0$, $(1,0,1) \rightarrow f_2=0$, $(1,1,0) \rightarrow f_2=1$, and $(1,1,1) \rightarrow f_2=1$. The two truth tables are identical, demonstrating that the two functions are equivalent. The NPTEL logo is visible in the bottom left corner of the slide.

Now rename the variable in B_x to their primed version, now what happens? Say if I am going to give your function say f is equal to $a b + a' c$. Now if I going to write another function, say I am going to say this is your f_1 and I am going to say write another function say f_2 ; I am going to say that, $x y + x' z$ or $x \bar{y} + \bar{x} z$. I can say that this is basically, negation of that thing; so I am saying that $x \bar{y} + \bar{x} z$.

Now if you look into this particular two function f_1 and f_2 , now I can realize sorry you can realize, that these are equivalent function because what will happen the variable a is correspond to variable x variable b is correspond to variable y and variable c is correspond to variable z ? That means they are having the name of the variable is different that means, you can say that we are just renaming this particular variable from $a b c$ to $x y z$. If you look into the evaluation, if any evaluation say x is equal to 1, a equal to 1, b equal to 1 and c equal to 1 what about functional value you have going to get for f_1 ? With x equal to 1, y equal to 1 and z equal to 1 we are going to get say evaluation for f_2 . For all combination, we are going to get the same evaluation, so that is why we are going to say that f_1 and f_2 are equivalent or they are representing the same function. So these two function f_1 and f_2 basically same, they do not have any different only we are rename in the variable $a b c$ to $x y z$.

Now similarly, in our transition system also now what will happens? Say for f_1 I am going to get an BDD, so this is your f_1 . If you look for the BDDs for f_2 , we are going to get the same BDD here will be no difference because; they are representing the same one. Only the variable will be rename now if this is you're a , now it will be rename to x similarly, if next level if I having the variable b , then that will be y one. Now similarly, what will happen in our B_x , this is the OBDD representation of our set of state x with this particular variable x_1, x_2, x_3 . So, B_x is your this thing BDD representation of a sub set of state, sub set of states and say that variables are your x_1, x_2, x_3 like that x_n .

(Refer Slide Time: 16:39)

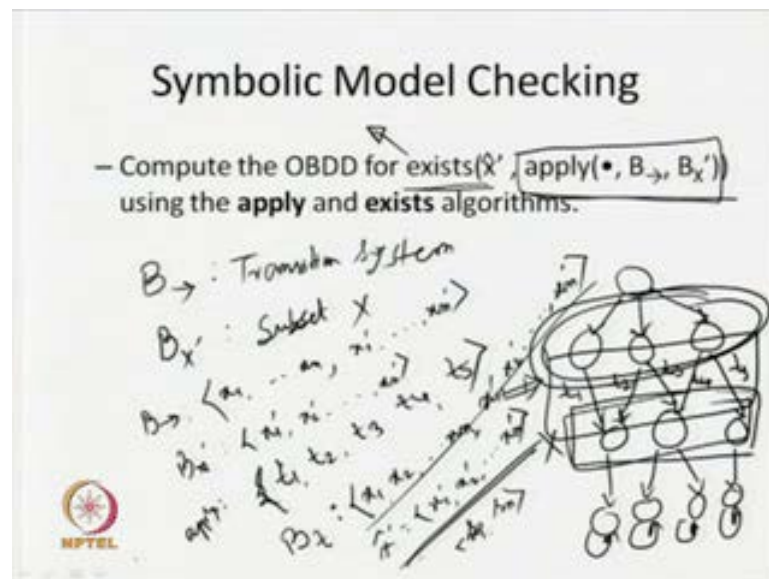


Now B_x prime, what we are having? We are going to have the same BDDs but renaming this particular variable x_1 prime, x_2 prime, x_3 prime, x_n this is the scenario that we are having. Now I will see this thing, say if I am having a transition system something like that, so this is the sub set x , the set I am having say s_4, s_5, s_6 . Now, what basically happen, say we are having if I am having a transition s_n to s_m , then we are going to represent this transition with this particular order of your s_n, s_m .

Now, when I am having this particular sub set, so I am just saying that this is the B_x is representing the this particular sub set s_4, s_5 and s_6 . Now say B_x they are representing with the state variable, now I am just reaming those particular variable in B_x prime to their primed version x_1 prime, x_2 prime something x_n prime. Now what is the basic

basically, if you look into those particular transition because our aim is to find out the predecessor state that means; these are basically next state variable and we are going to find out what are the present state variable. That is why we are renaming this particular variable to their primed version, just capture those particular transition. This is our basic aim, so after renaming the variables we are getting the same structure as OBDDs; so whatever OBDDs we have the rename structure will remain same on the we are renaming those particular variable.

(Refer Slide Time: 18:27)



So after renaming the variable we are getting Bx' , now what is the next state? Compute OBDDs for $\exists x'$ had I should say, $\text{apply}(\cdot, B \rightarrow, Bx')$ using the apply and exists algorithms. Now in this particular case what happens you just see that, if I will concenter this particular relation, say simple transition system; now say that I am I can or set of completion I can complete it say this is a sub set x . Now we are getting $B \rightarrow$ as the OBDD representation of my transition system and Bx' is the representation of this sub set x . Now we are having this representing this particular sub set with their prime version of the variable.

Now in B transition the BDD of the transition system, we are having all the variable x_1 to x_n and x_1' to x_n' all the variable are available in your B plane and your Bx' we are having the prime version of the variable. Now when I use this particular apply operation the dot operation, what basically dot? It is going to give me

the intersection of two sets. Basically, already we have discussed, we have mentioned that this if you apply this particular dot operation it is going to give me the intersection of two sets.

Now I am using tools BDD, this is one is representing some states and second one is representing the after transition system. Now when we apply this particular dot operation, it is going to give me a command or intersection of these two particular state, then basically what we are going to get after applying this particular dot operation, we are going to get basically those transition say this t_1, t_2, t_3, t_4, t_5 . So it is going to give me, say in my entire BDD representation of the entire transition system B arrow, I have all the transition; now I have just intersect this particular transition all transition with this particular subset.

Now whatever we are getting, we are getting the transitions that are all coming into this particular x , since other transition which simply go over. So this is a intersection operation we are doing, so basically we are getting after performing this particular apply operation after apply what we are getting? We are getting those particular transition only t_1, t_2, t_3, t_4, t_5 we are getting this five transition because these are the five transition, these are the intersection of these to particular sets. Now when whatever is the result you are getting over here, we are getting the transition; that is coming into all the transition coming into this particular subset x .

Now in this particular BDD whatever resultant BDD we are getting say, I am going to say that this is your BDD say only transition that it is having, so these particular BDDs having all the variable x_1, x_2 to x_n and your prime version x_1', x_2', x_m' . So we are getting those particular transitions, which are coming into this particular set x and these particular transition will be represented by these all those particular variable, the state variables along with your next state variable. Now whatever BDD we are getting, now what we are doing it; now we are using this particular exist algorithm and what we are exist? We are making B exist x' had that means; we are taking this particular factor x' had x_1', x_2' like that x_n' .

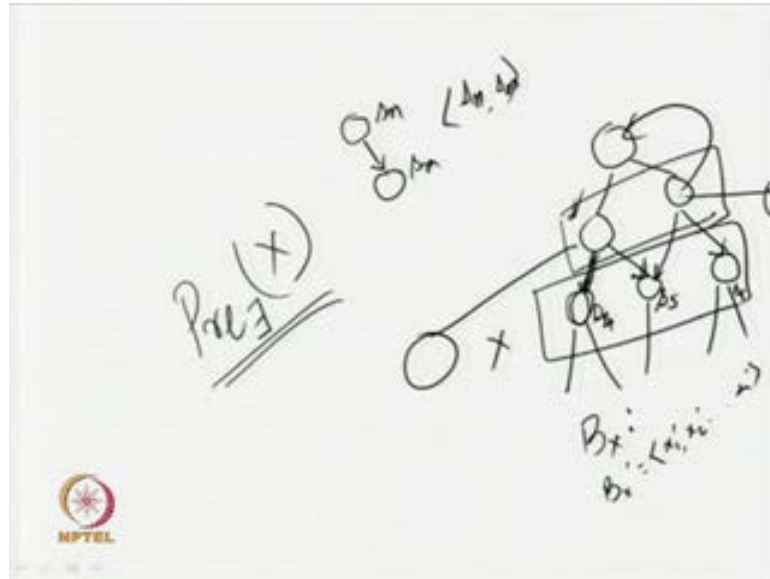
So what we are doing basically, whatever result and BDD we are getting, we are making it independent of these particular prime version of that variables. You just see what we doing? We are first doing the apply operation and we have applying this particular dot

operation, we are getting the intersection, we are getting those particular transition which are of our interest that means; the transition which are coming to this particular set x . Now from that what happens? Now we are making it independent of the next state variable. So when we have making it independent of this particular next state variable so what happens? Now, these transitions are having all the variables x_1 to x_n and your prime version x_1 dot to x_m dot. So this is basically representation of this particular state from present state to next state algorithm.

Now we have making it independent of those particular next state variables, so it is having all the variables; now we have making it independent of next state variable. So in this particular case what we are getting? In the resultant BDD, whatever final BDD, we are getting now it is independent of next state variable. So eventually, we are getting the representation of those particular states on there, just try to visualize it and try to understand it; that we are getting these particular states on them that representing those particular states and these are represented by the state variable of present state variable.

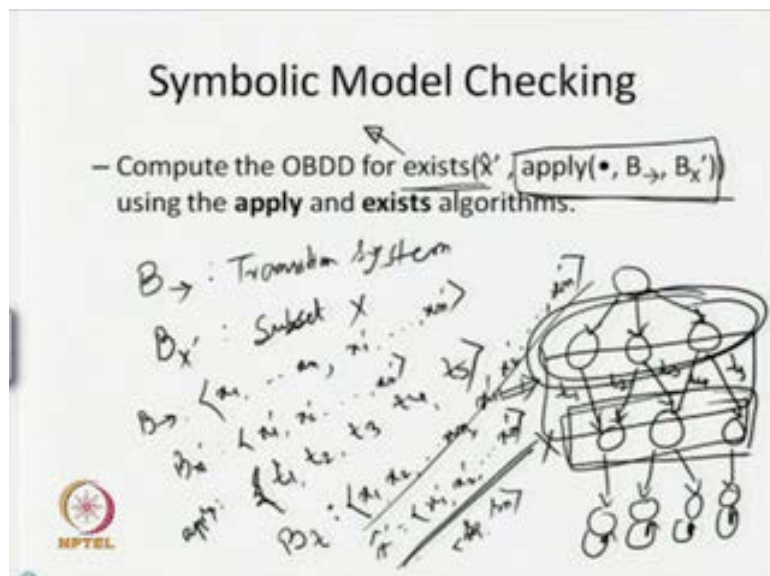
You know, just see that what apart the ROBDD or OBDD written by this particular operation, it is going to give us the states or OBDD representation of a sub states from where all the transition are coming into this particular sub set x . So this is the way we are evaluating, so that means we are getting a resultant BDD from that resultant BDD of a subset and from those particular all the state from this particular sub state all transition are coming into this particular sub state x .

(Refer Slide Time: 25:34)



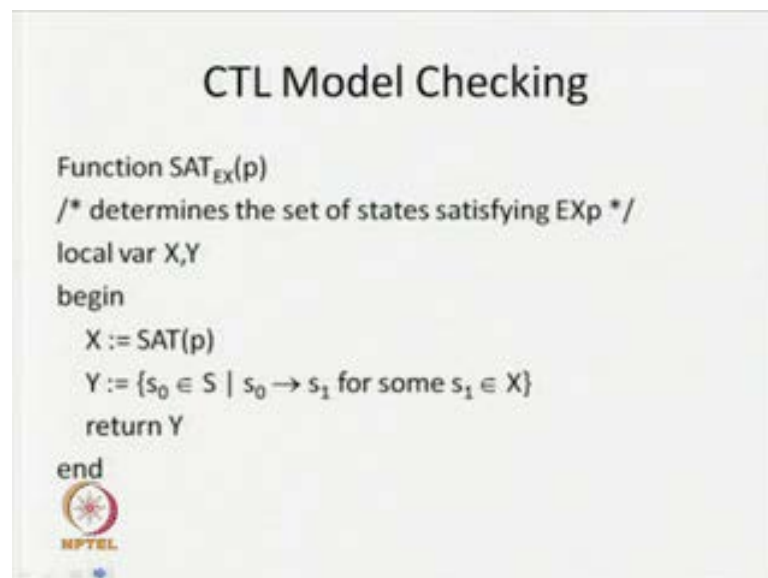
So this is the way, we are calculating the pre all the predecessor state. So, if all the predecessor state we are getting in, so predecessor they are exist x will also be the same state because we are getting all the states from where the transition all the transition are coming into this particular subset x. So that means at least there will be one transition which is coming into this particular set, so it is going to give me the pre they exist x. So if I am having some more transitions something like that, spiel these two states will come into as a resultant state. Now, since at least some of the states are coming into this particular x that means; this is also candidate for your p they are exist x.

(Refer Slide Time: 18:26)



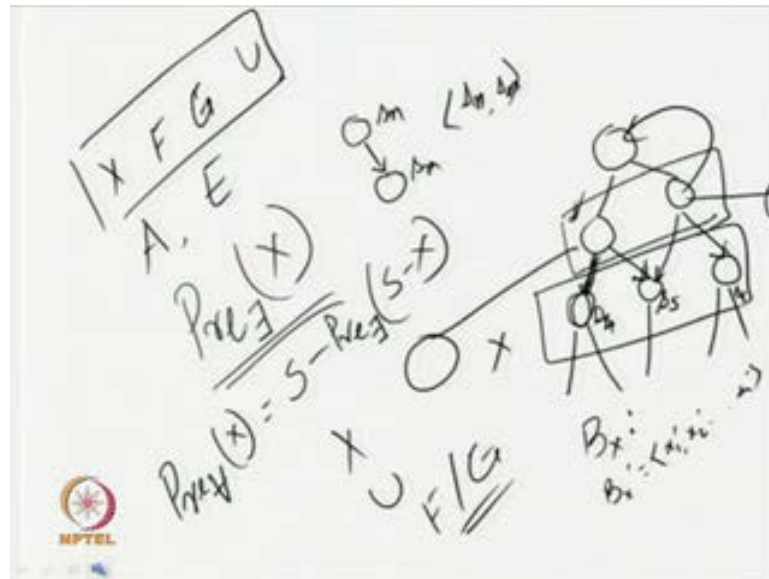
So with the help of this simple operation, with this particular two stapes, first we are renaming the variable of B x and then we are applying this particular operation or using this apply operation and exist operation to get the predecessor state and whatever predecessor state, we are getting all those states will give us a pre they are exist x. And once we are having the pre they are exist x, we can variable calculate the pre for all X, already we have seen this is nothing but s minus pre they are exist s minus x. So now we are having a method to find out pre they are exist X and once we can calculate pre they are exist X, we can calculate pre for all X also. Now this is basis or this is the basic requirement of our symbolic model checking algorithm.

(Refer Slide Time: 27:13)



Now in this particular case, now we are going to see, how we are going to implement those particular symbolic model checking algorithm.

(Refer Slide Time: 27:18)




Already you have seen that, we are having, we have discussed with four temporal operator; next state feature, global and until and this four temporal operators are predictors always preceded by the path quantifier A and E that means; in all path and they are exist a path. So we are getting a different combination but already we have seen that, if we get three operators also, that will going to give me a complete set of operators. So in that particular complete set of operator, I need the next state operator, I need the until operator, I need either feature or global any one of this two. So either all A or E x whatever it may be so we need three operators and already we have discuss the algorithm for those particular three operators, that complete set of operators. Now we are going to see, how you are going to implement those things symbolically that means; how we are going to implement symbolic model checking algorithm.

(Refer Slide Time: 28:20)

CTL Model Checking

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

EXP




So path operator is your we are talking about E X p that means; they are exist a path where next state p is true. This is the simple procedure what we are talking, first we have going to take all the states where p is true, so satisfying the p is going to return all the states where p is true. So, we are going to collect those state in such a whether from those particular states, we are having a transition to s 1 such that, s 1 belongs to this particular set X. So this is basically, the evaluation of E X now this procedure we can implement symbolically.

(Refer Slide Time: 29:02)

Symbolic Model Checking

EX(B_φ):
B_φ:OBDD for set of states where φ is true.
// Analogous to X := SAT (φ):
B_→:OBDD for transition relation.
Return Pre_→(B_φ). *// Analogous to Y := {s ∈ S | exists s', (s → s' and s' ∈ X):*

EXP
φ → X

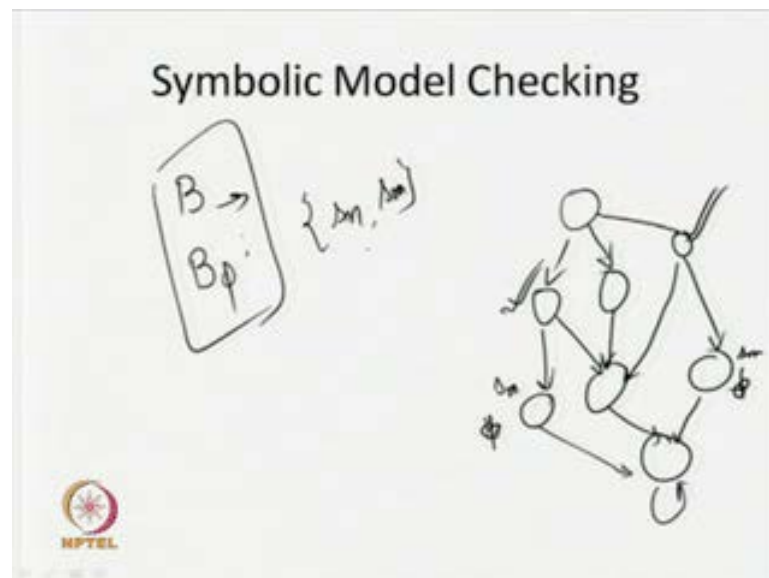


Evaluation of Pre_→(X)

So how much symbolical model checking algorithm so look like, you just see that $E X B \phi$ what is $B \phi$? It is a OBDD representation of the state of states where ϕ is true because when we are going to evaluate $E X \phi$, we must know the set of state where ϕ is true. We know the set of state where ϕ is true and we are going to represent those particular set of state with the help of n BDD and we are going to said that, $B \phi$ is the BDD representation of those particular set of state, where ϕ is true. So this is analogous to X assign, we are assigning the set of ϕ where the set of state ϕ is true to this particular x . So this is basically, analogous to x is equal to set of p .

Now what happens and B arrow is the OBDD representation of our state transition system. Now whatever variable we are ordering, we are having for B arrow that same variable (()) we should maintain for $B \phi$ also. Now from that what will happen? Simply you evaluate pre they are exist $B \phi$, now we are getting all the set, this is analogous to this particular set y is equal to you are going to collect all the set from s from s ; so that we are having a transition of s to s dash, where s dash belongs to this particular X . So this is the way that we are going to have, so we simply use this particular pre they exist x .

(Refer Slide Time: 30:28)

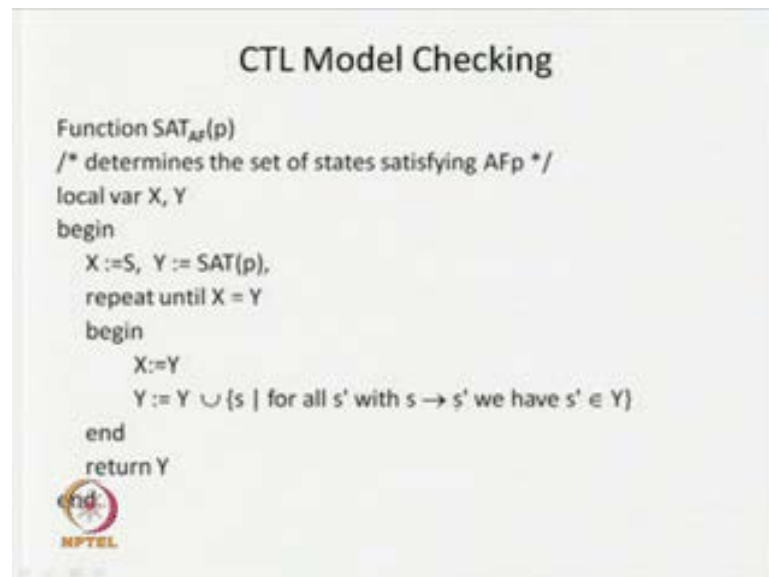


Now what basically just see that we can think about something like that, you say this p is true over here; then first we are going to get B arrow, which is the representation of this particular whole transition system. Then we are going to get say $B \phi$ the set of state,

where ϕ is true; say these are the two states where ϕ is true. So I am going to say that s_m and s_n , so this is $B\phi$ is nothing but the BDD representation of your s_n and s_m .

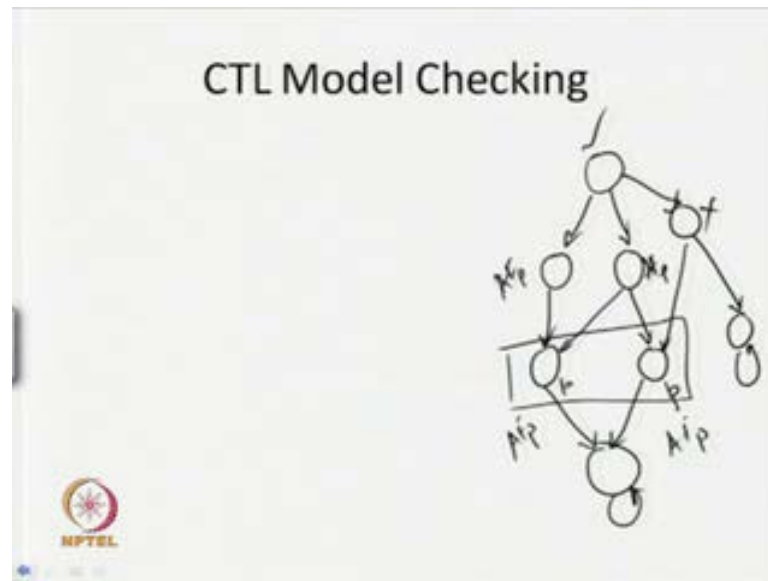
Now we are going to look for the all predecessor state, so if you can going to calculate the predecessor state. Basically, these two states will come as forward this evaluation, we can pre they are exist $B\phi$ and these are the states where $EX\phi$ is true. So we are going to give the input as our $B\phi$ and B transition and from that we are going to evaluate this particular predecessor state. Just see that, pre they are exist X or pre they are exist $B\phi$ is going to give me the set of state where $EX\phi$ is true. So this is one algorithm, this is a very simple one.

(Refer Slide Time: 32:04)



Now second one, we are going to say that state AFp . So determine a set of states where AFp is true, already we have discussed these algorithms. So we are going to perform with operation wherever p is true, we are going to say that AFp is true. This is s for a semantic, that we have discuss our semantic says that, present state includes the future behavior and after that we are going to look for all the states from where we are having a transition to those particular state. So we are going to collect all state, so that we are having s this some. We are having a transition of s to s' , where s' belongs to y look like.

(Refer Slide Time: 32:43)



So this is basically what happens? We can say something like that $A F \phi$ in all part in future ϕ may be say, if I am having something like that, then say if p is true over here. Just say that you in one state I am going to take, then all state p is true over here also; then I am going to this particular sub state, we are going to calculate the previous state, since all the transition that are coming from this. So these two states will be marked with your a $F p$, now we are going to perform the backward traversal algorithm.

So in this particular case, say all the transitions are coming to this particular sub state x . So $A F p$ will be true over here, $A F p$ will be true over here, then if I am going to look for this one, then $A F p$ is true for two states but these state is not $A F p$ is not true; so these state will not given to us. So we are going to repeat this particular procedure. So that is why we are saying the collect some more states. Then again see go over and see where x is equal to y or not, some point of time that means; we have not adding any more states, then we are going to terminal this particular procedure.


(Refer Slide Time: 34:15)

Symbolic Model Checking

AF(B_ϕ):
 B_ϕ : OBDD for set of states where ϕ is true. // Analogous to " $\neg Y := SAT(\phi)$ ";
 B_\rightarrow : OBDD for transition relation.
 B_X : OBDD for all states of the system. // Analogous to " $X := S$ ";

repeat until $B_X = B_\phi$ // Analogous to "Repeat until $X=Y$ "
 $B_X := B_\rightarrow$ // Analogous to " $X := Y$ ";
 $B_\phi := \text{apply}(\rightarrow, B_\phi, \text{Pre}_\rightarrow(B_\phi))$ // Analogous to " $\bigcup_{s \in S} \{s \mid \text{for all } s', (s \rightarrow s' \text{ implies } s' \in Y)\}$ "
end
return B_ϕ

B_ϕ
 B_\rightarrow
 $B_X =$
 $\text{Pre}_\rightarrow(B_\phi)$
 $\text{Pre}_\rightarrow(X) = S \cdot \text{Pre}_\rightarrow(S \cdot X)$



So this is the same procedure, we can use, now we can apply our symbolic model checking algorithm. So, what we are having? Say B_ϕ is the OBDD representation of the set of state where ϕ is true. First we are going to start from those particular set of state, where a given formula is true like that; we are starting from this particular sub state and now going to perform the predecessor operator and going to find out what are the states where $AF \phi$ is true or $F \phi$ will be true.

So first we are going to take the OBDD representation of the set of state, where ϕ is true and B_\rightarrow is the OBDD representation of our entire transition system or you can say that, it is OBDD or it may be a reduced OBDD also. So we are starting with these two OBDD, B_ϕ and B_\rightarrow , then we are going to take one particular this things B_X . Basically it says that, these are say all states of the system, I am just representing it with the help of this BDD. So we are going to take one BDD, where it is going to represent the entire states. It is not the state transition system but this is the all the states OBDD representation of all the states. So this is basically analogous to or I am saying that, miscellaneous taking that x is equal to your s , I am just express.

Now we are going to repeat this particular procedure, the way we are doing in the previous case that, repeat until x is equal to y . So we are going to repeat this thing, until that B_X is equal to B_ϕ , so B_ϕ is assigning to B_X that means; we have keeping the previous information. Now we are going to use this particular operation, we are going to

use this particular apply plus because what we have to do? Whatever we are having, we are going to perform the union with some new more states.

Now what are the new states is coming? So this basically union, union is going to use by this particular plus operator at $B \phi$ is the set of state with correspondent to this particular y set of state, where ϕ is true and now this is your in all path in future, so that means we need pre for all. So this is basically, we are going to calculate pre for all $B \phi$ and how to calculate it? Already we have seen that we are having procedure to evaluate pre there exist $B \phi$ or x . So we will use this particular relationship to get pre for all x ; so that is why I am just crisply, I am writing over here, I am not writing a entire expression. So we are going to look for all the predecessor state from where all the transitions are coming into this particular sub set, where ϕ is true.


So I have getting a new state then again I will go back and repeat now I am going to see what is the previous set of state we have, now whether it is equal to $B \phi$ or not. So new $B \phi$, if there equal then will terminal, if it is not equal that means; you have where at some new more state like, these particular case say you have where at some new more state. Now we have to check get the predecessor of those particular new states, so we will again enter to this particular procedure. So just see that, this is a simple conversion of our model checking algorithm to the symbolic model checking algorithm, where we are using OBDDs; ordered binary decision diagram to represent the entire system and represent the set of states, where a particular formula is true and applying this particular algorithm.

(Refer Slide Time: 37:54)

CTL Model Checking

```
Function SATEU(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
```

$E(p U q)$

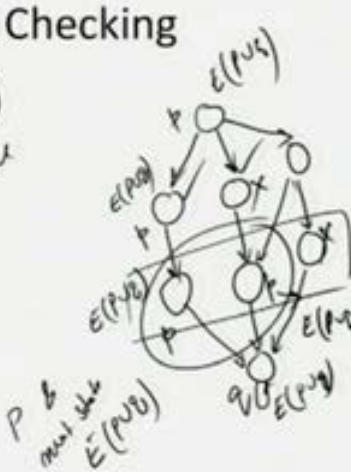


So similarly, we need one more procedure, where we are having set $E U p$ until q . So E exist a path, p remain to until q becomes true, so $E U p$ until q . So what is the procedure already we have discussed it, so what we have to do? We have to look for p remains to until p is true. So this is the expression or this is the evaluation we did actually, what is the w ? w is the set of state where p is true that means; you have to collect those particular states and intersection with this particular new state that, we are having a transition from those particular state and p must be true.

(Refer Slide Time: 38:32)


CTL Model Checking

$E(p U q)$
 q : true



$P \ \& \ q \ \text{must be true} \ E(p U q)$

$q \ U \ E(p U q)$



So basically you just see the transition, this I am drawing it something like that. So if you are having this thing, then what will happen? First we are going to look for those particular states, where we are looking for $E p \text{ until } q$. So we are going to look for those particular state, where q is true; since q is true over here. So I can say that, $E p \text{ until } q$ is true because it is as for our semantics, that present into the future. Now from this particular state what happens? We are going to look for a predecessor state, so we are going to get this particular pre predecessor state.

So in this particular case, p must be true and the next state $E p \text{ until } q$ must be true. So this is the scenario, that we are having w , it is where is the set of state, where p is true intersection with those particular states, where in next state q is true. So in this particular case what will happen, in the state whatever we are getting over here; then we will find that, this particular state p is true and next state $e p \text{ n thinking}$ is true. So you can say that, $E p \text{ until } q$ is true; here also we will get that $E p \text{ until } q$ is true, but in this particular state $E p \text{ until } q$ is not true because p is not over here.

Now I am getting some more states, so we are going to look for a predecessor of those particular new states. So I am going to get these two state again with the same property, same constant that p must be true over here, say w intersection of those particular things; we must have transition to this some state of this particular state y , where y is the new set we are constructing. So again we will find that $E p \text{ until } q$ will be true over here but it will not true at that particular point; now when we come back to this thing again $E p \text{ until } q$ will be true. So this is the way that we were evaluating $E p \text{ until } q$. Now this same procedure we are going to implement symbolically.

(Refer Slide Time: 41:08)

Symbolic Model Checking

EU(B_{ψ_1}, B_{ψ_2}):

B_x : OBDD for all states of the system. // Analogous to $\neg X = S$

B_{ψ_1} : OBDD for set of states where ψ_1 is true. // Analogous to $\neg W := SAT(\psi_1);$

B_{ψ_2} : OBDD for set of states where ψ_2 is true. // Analogous to $\neg Y := SAT(\psi_2);$

B_{\rightarrow} : OBDD for transition relation.

repeat until $B_x = B_{\psi_2}$:

$B_x := B_{\psi_2}$ // Analogous to $\neg X := Y$

$B_{\psi_2} := \text{apply}(\cdot, B_{\psi_2}, \text{apply}(\cdot, B_{\psi_1}, \text{Pre}_{\rightarrow}(B_x)))$

Analogous to $\neg Y := Y \cup (\exists x \in S \text{ exists } x; (x \rightarrow x' \text{ and } x' \in Y));$

end

return B_{ψ_2}

Handwritten annotations: "Predecessor" (with arrow pointing to Pre_{\rightarrow}), $E(\psi_1 \cup \psi_2)$ (written vertically), and $E(\psi_1 \cup \psi_2)$ (written vertically).

Now what we have to see? Again similar to the preparation, now it is basically E psi 1 until psi 2; so we need two BDD s to represent the set of states, where psi 1 and psi 2 are true. So we are using starting with these two BDD, B psi 1 and B psi 2; B psi 1 is the OBDD representation of the set of state, where psi 1 is true and B psi 2 is the OBDD representation of the set of state, where psi 2 is true. So these are the two things we are getting, then we are taking at the entire state transition system B arrow, this is the OBDD representation of the entire transition system.

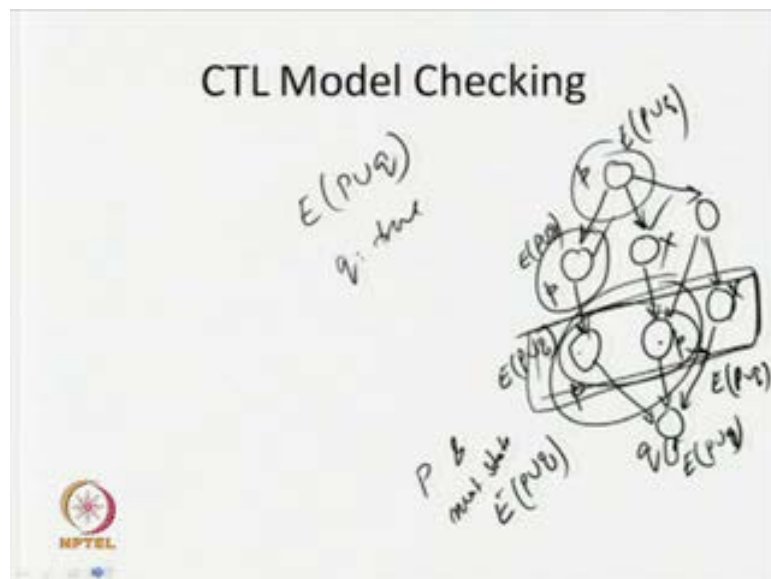
Now as for our semantics, wherever psi 2 is true, then E psi 1 until psi 2 it is also true on those particular state. So I am starting with these particular psi 2 and we are saying that initially, making B x is equal to psi 2. So initially, I can construct B x as the entire states all the states of my states space. So initially, wherever psi 2 is true, E psi 1 until psi 2 will be true; so we are just keeping this particular assignment. So now we are going to compare these things, we are going to repeat this particular look; now loading all those particular states psi B psi 2 to B x, just renaming it, then we are again using this particular expressions.

So what happens? These particular portions will give me this portion. So pre they are exist X, it is giving the all the predecessor state in predecessor state psi 1 must be true. So we are doing this particular dot operation, that is the inter section that means; this is the set of state, all predecessor state. Now B psi 1 is the set of state, where psi 1 is true

that means; this is nothing but the states space representation of this particular w , w is the set of state, where ψ_1 is true.

So we are taking the intersection that means; it is going to give me those particular two states in my previous example, say it is returning my these three behavior of predecessors state but after doing the intersection, I am going to get these two states on this. So this is the operation we are performing, so these are the states where it is true and now I am going to use this apply plus that means; I am going to do the union operation that means, first we are collecting the set of state, where ψ_2 is true and I am say that ψ_1 until ψ_2 will be true all those particular state. Now we are collecting new more states, so we are going to use the union operation we adding them to in this particular state. So that means; this union operation has been perform by these particular apply plus operation.

(Refer Slide Time: 44:18)



Now we are adding some more states, where this particular $E \psi_1$ until ψ_2 is true like that we are adding these two states, now we will again if we this particular procedure to get the state and this state.

(Refer Slide Time: 44:26)

Symbolic Model Checking

EU(B_{ψ_1}, B_{ψ_2}):

- B_S : OBDD for all states of the system. // Analogous to $X = S$
- B_{ψ_1} : OBDD for set of states where ψ_1 is true. // Analogous to $W := SAT(\psi_1);$
- B_{ψ_2} : OBDD for set of states where ψ_2 is true. // Analogous to $Y := SAT(\psi_2);$
- B_T : OBDD for transition relation.

repeat until $B_{\psi_1} = B_{\psi_2}$

$B_{\psi_1} = B_{\psi_2}$ // Analogous to $X = Y$


$B_{\psi_1} = \text{apply}(T, B_{\psi_1} \cdot \text{apply}(T, B_{\psi_2}, \text{Pre}_T(B_{\psi_1})))$

Analogous to $Y := Y \cup \{W \mid \exists x \in S \exists x' (x \rightarrow x' \text{ and } x' \in Y)\};$

end

return B_{ψ_1}

$E(\psi_1 \cup \psi_2)$
 $E(\psi_1 \cup \psi_2)$



So this is the way that we are doing. So after adding this particular two states we perform go into this particular look, I can we are going to check whatever is the new states, we are getting whether it is equal to our previously collected state or not. If they are equal then will terminal, if they are not equal, then again I am going to know this same procedure. This is similar to repeating these particular states.

(Refer Slide Time: 45:16)

$X \quad F \quad G \quad U$
 $CTL \quad A \quad E$
 $AX \quad AF \quad AG \quad AU$
 $EX \quad EF \quad EG \quad EU$

$EX \phi$
 $AF \phi$
 $E(\psi_1 \cup \psi_2)$

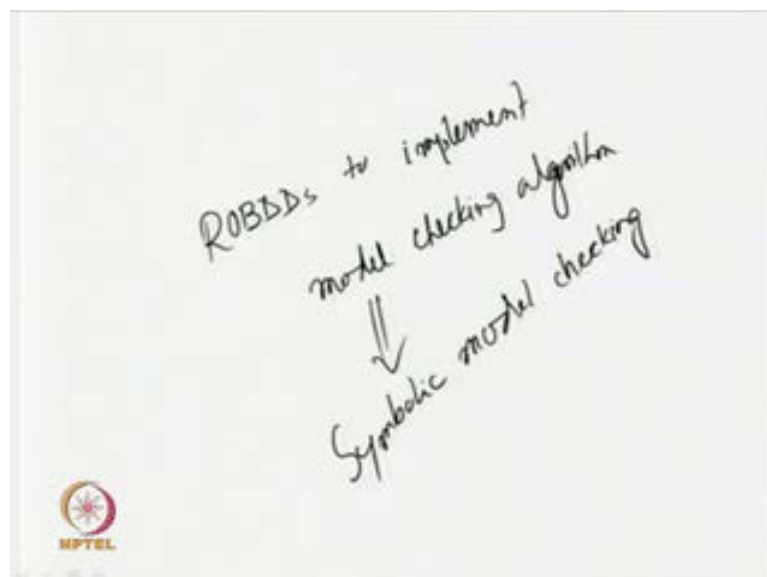


So in this way we are calculating this, so you just see that, all this true operator what are the operators that we are talking about here, three operators we are talking about one is

your. So we have talking about the operator $E X \phi$, $A F \phi$ and $E \phi_1 \text{ until } \phi_2$. So since we are having the procedure to evaluate these three operators; now we can evaluate the other operator also because we have seen that these are the complete set of operators. So we need basically eight operators, temporal operators that we have discuss till now. We are having four temporal operators, next state future, globally until and with respect to this particular four temporal operators, we are going to get eight CTL operators with path quantifier A and E, in all path and they are adjustable that means; $A X$, $A F$, $A G$, $E U$ and similarly, $E X$, $E F$, $E G$ and sorry this is $A U$ and $E U$

So out of that, if we are having these three procedures for these three operators, others can be evaluated. So you just see that, we are using ROBDDs to evaluate these three procedures; now we are getting a symbolic model checking algorithm and what is the advantage we are getting? We are representing the OBDD the centre states phase with a help of ROBDDs. And in most of the cases, we have found that we are going to always get a compact representation. Again I would like to mention that, the size of ROBDDs depends on the ordering of the variable but to get the exits ordering is a hard problem, already I have mention all those issues. Now we have seen that, we have got that model checking algorithm or symbolic model checking algorithm for our this thing.

(Refer Slide Time: 47:20)



So we use ROBDDs to implement model checking algorithm and what our model checking algorithm we are getting? We say this is your symbolic model checking, we have seen this particular things.

(Refer Slide Time: 47:57)

Tools

- CUDD
 - CU Decision Diagram Package
 - University of Colorado at Boulder
- nuSMV
 - Extension of SMV, the first model checker based on BDD
- SPIN
 - LTL model checker developed at BELL labs

BDD

NPTEL

Now you just see that, now we are having enough idea about our model checking however what we can do with the help of model checker? What cannot be done, all those things we have studied. Now in this particular case, I would to mention some tools; one of the tools is your CUDD, this is basically a package for your decision diagram or say binary decision diagram. So this is a package for your BDDs, this is CUDD it is developed in university of Colorado at boulder. So universe the Colorado at Boulder, they have developed this particular package. This is a clear everywhere try can you purpose and you would you can use it now what you can do? You can download this particular cut package from university of Boulder university of Colorado at Boulder and use this particular package to construct BDD to manipulate the BDDs, use it is having all the required algorithm.

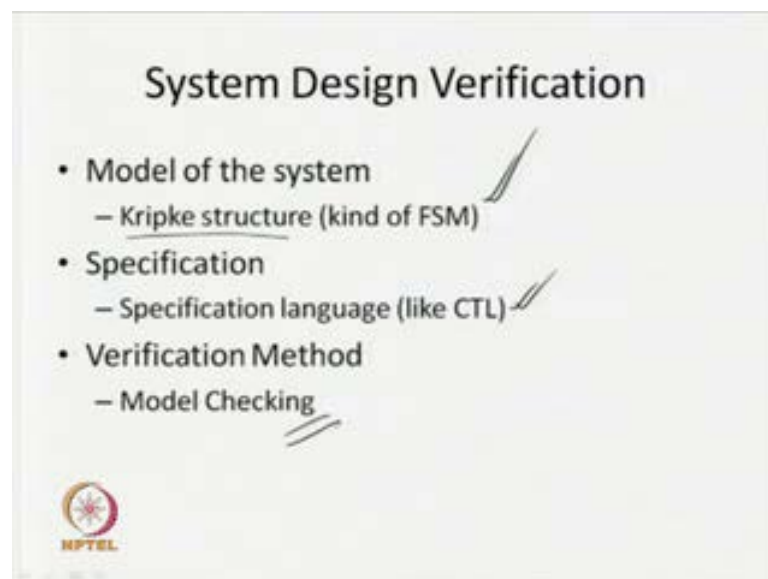
Another tool I am going to just mention over here, which is your nuSMV, this is your symbolic model verify. It is the extension of SMV symbolic model verifier, which is the first model checker based on your BDDs. It was developed by Ken McMillan who was a student of your adamant class, who is Parnell of this particular model checking procedure during his tenure it in CMU, he develop this particular SMV. Now, it is

having some modification extension and release as your nuSMV. So this is a symbolic model verifier, whether is in a BDD to represent entire system and do this. So with the help of this thing, we can model our system or we can look for property verification, it is handles Parnell's also.

Another one I can just mention over here, it is SPIN; this is a tool developed at your bell labs this is initially the LTL model checker. So we can already we have mention about what is LTL which is, so we are having a LTL model checking. Also now what you can do? You just try to download this three package and try to work with this thing. So working is also very simple, what you need to do? You have to see; you have to look into the syntax how we are going to give the input to this particular tool.

So you have to know the syntax of nuSMV, how to provide the input and you have to know the input syntax for the SPIN verifier. They using the particular language call Promela, so that you need to knew the, know the syntax of this particular formula language to give the model of your system and after that after specifying the property; it is going to check whether the property is true or not. So this is the way, that we are doing it. So what happens now? You just try to for I am just giving this information, if you are interested you can download those package and try to work with those package.

(Refer Slide Time: 50:54)



Now what we have seen in this particular case, say this is basically we are discussing about your system design verification. So we are having a design and we after coming

with up a design, we are try to verify or check whether my desire properties are true in this particular design or not.

So in this particular case what we have discussed? Or what we need? Basically, we have to come up with the some model of the systems. Say if I am going to design something, say simple one I can talk about that you are going to design for your. So traffic like controller, we have to come up with a model and this model basically, we are representing with the help of Kripke structure which is a kind of your FSM, finite state machine.

After that we have to give our specification or the property death not be satisfied by my model and here in this particular course, we have discuss one specification language which is call CTL, computational tree logic. We have discuss with this particular CTL, we can give the specification with the help of this particular CTL language and after that we need the verification method by which we are going to check, whether given specification is true in the model that we have given or not. So for that, we have used this particular model checker or we are using this model checking algorithm.

(Refer Slide Time: 52:15)

The slide is titled "System Design Verification" and contains the following content:

- Model Checking Algorithms (CTL)
 - Polynomial algorithm
 - Method can be easily automated
 - It provides counter example
- Problem with model checking
 - State space explosion problem
- Symbolic Model Checking
 - Use of OBDDs to content the state space explosion problem

Handwritten notes include "CTL" next to the first bullet point and "n n p" next to the second bullet point. The NPTEL logo is visible in the bottom left corner.

Now after that, when we look into the model checking algorithm, what we have find? Or we have observed that, we are going to get an polynomial algorithm for model checking algorithm. So this is the beauty of your model checking algorithm, if I talk about a model checking algorithm, particularly I should say that, in particularly should say that this is

your CTL model checker. For CTL model checker, we are getting an polynomial time algorithm, so which can be manageable, which can be handled and the method can be easily automated.

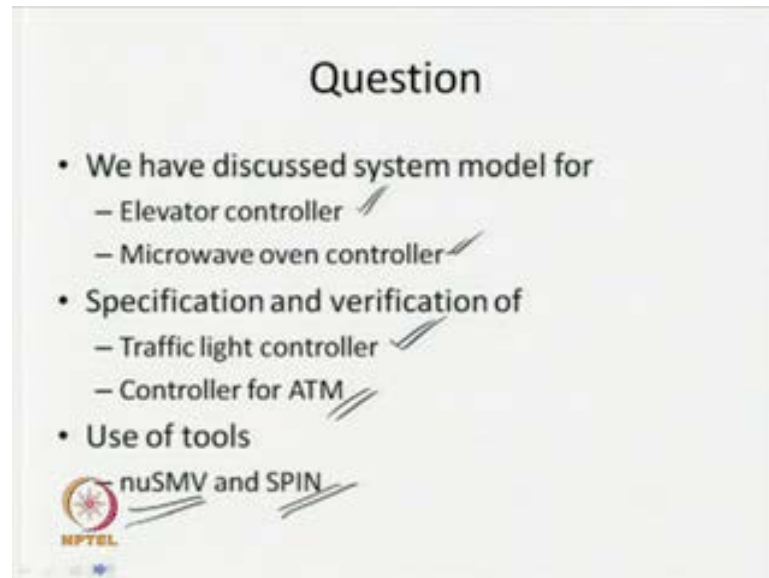
Now since we are having an algorithm, which your polynomial algorithm, so it can be automated that means; we can have an automated tool. I am already I have mention about some tools like nuSMV, SPIN like that. And another advantage of this particular model checker is, it provides counter example. So if I am giving a model and I am giving a specification of property, now using this particular model checker, then what will happen? If the property is true, it is going to say that; yes, this property is true in your model you can proceed but if the property is not true, it will give me give a counter example, it will give me an execution trace and it will say that if you follow this particular execution trace, then you are given properties falls.

So this is a some sort of feedback mechanism, it is giving some feedback to the designer team; now designer team can concentrate on those particular execution trace and they can fixed the bug. So this is the advantage about this particular model checking algorithm. Now what we have seen? We have seen that, we are having some problem with this model checker. What is the problem? The problem is takes place expression problem. Already I have mention that, the states place of the system, of the model that we are going to get is exponential with respect to the number of state variables that we have. Already I have mention that, if I am having n state variable, the number of possible states will be a 2 to the power n , if n is increase by 1 then it will be 2 to the power n plus 1 which will be exponential in nature.

How to contain? How to restrict this particular states place expression problem? Or how we are going to work with a bigger system? So for that, we have seen how compactly it can be represented that transaction system can be represented and how it can be represented? What we need? And we have seen that, the data structure BDD can be used or in particular ordered BDD can be use to represent the state space diagram and most of the time we are going to get a compact representation for those particular states space. So by using OBDDs, we are coming up with another model checker and we call this is your symbolic model checker.

So in case of symbolic model checker, we are using OBDDs to represent our states space and we are somehow containing the states space expression problem, we are restricting the explosion with respect to this particular states space. So these are things that, we have discussed in this particular part.

(Refer Slide Time: 55:09)



The slide is titled "Question" and contains the following content:

- We have discussed system model for
 - Elevator controller ✓
 - Microwave oven controller ✓
- Specification and verification of
 - Traffic light controller ✓
 - Controller for ATM ✓
- Use of tools
 - nuSMV and SPIN ✓

At the bottom left of the slide, there is a logo for NPTEL (National Programme on Technology Enhanced Learning) and the text "nuSMV and SPIN".

Now as I now question, what I am going to say that, already if you look back our lectures, then what happens? We have seen how to design an elevator controller. We have seen or we have discuss a simplified model of a microwave controller, so these two we have discuss in our course of our lecture. Now what we can do? Now since you have the model, you know what are the properties that you need to verify, already I have mention something along with that very simplified the one of very simple one, that we have discuss about your Michel exclusion problem for our said resources. So these are the things that we have discussed in our course of lectures, now what we can do? That I have mentioned about some tools like your nuSMV or SPIN.

Now you try to model this system on those particular tools, model this system means what happens? Already I have or we have discussed about the model. Now you just try to convert this particular model to the input language of you SPIN or nuSMV and you see, how we are going to write the specification for those particular tool and check those particular properties. Just as an example as a question I am giving you, you download

those particular tools, install in your machine and try to check how those particular model checkers are going to work.

Now, I am giving some other problems also, say what I am talking about that traffic light controller, say already in many a time I have discussed or I have mention about this particular thing. Now you take some complicated system, complicated thorough network and try to design a controller for this particular traffic on this particular junction. So that means; we need a controller, so you try to come up with a design, come up with a model and come up with the properties, that need to be satisfy this particular controller.

Another one, another system you can talk about ATM. Now it is most of you people are using that ATM, automated teller machine, so how it is going to work? You must know and if you try to analyze it, you will find that, you are all come up with a basic model. So you try to come up with a model of this particular ATM system also and after that after coming up this particular ATM model, then you try to find out what are the properties that need to be satisfy this particular model? And now use those particular model checkers to check those particular properties. So that is why I am saying that use of tools nuSMV or SPIN because these are freely available for convene purpose. Now you can come for, you can look for the designing or modeling of those particular system and see try to prove the properties on those particular system; in that particular case, what happens? You will be knowing how to use these particular tools also.

(Refer Slide Time: 58:04)



Now what we are looking into it, this is some sort of your designing of our digital system in this particular course, we are talking about design verification test of your VLSI system. Already I have mentioned or we have mention that basically, if you look into the design cycle, you have found, you will find this particular step. First you have to come up with the specification of the system, then we have to come up with an design; once we have come up with a design before proceeding further, we have to say that our design is going to work correctly.

So for that we have been coming up with this particular verification methodologies, we are going to check the properties in our model. Once we are coming with this particular, once you verify these properties and we are satisfy that yes, my system is going to work correctly, then we will go for implementation. After implementation then what we have to do? We will go for this particular testing, whether my implemented circuits are correct or not basically, it is going to find out the fabrication fault and next steps are next stage is your installation and marketing and after that, you have to maintain it or you have to go for the observation. So these are all design cycle steps that we have or seen or we have discuss that these are the steps where we required, when we go for this designable digital system.

(Refer Slide Time: 59:25)



And in this particular course, this is about the digital VLSI design. If you see that in this course, we are talking about the design of a digital system and if you look in to it, we

will find that, we are having three part of this particular course; one is the design issues, second is the verification issues and third one is the testing or testing of our design or testing of our VLSI design. So in this course, we are having these three issues. In the past module of this particular course, we have discussed about the design issues how will precede to design a system? This is the first part of this particular course and second part we are talking about the verification. So this is the second part that we are discussing till now in this particular part, we are talking about the verification issues.

So what we are doing over here? We are coming up with the design, so in while we go for this particular design we are coming up with a design, we are getting a model. After getting the model, we know what are the properties it must satisfy. So will come up the specification, we will apply some verification methodologies to check whether this property is two or not. So in this particular part, we are discussing about these particular verification issues, how we are going to do go for verification; and how what we are going to do and all processes and we have discussed and in this particular part, we have just introduce one particular technique call model checking. We have seen how we are going to use this particular model checker? What are the problems? And we have introduced about your symbolic model checker also.

And in the third part of this course is your testing, so in next part we are going to talk about this particular testing. One my fabrication is over now, how we are going to test the system, that before releasing into the my card we will said that, this is fault way. Basically in testing, what we are going to do? We are going to do about the going to find out the fault and we are going to release a fault free circuit to the market. So this is basically, post implementation of post fabrication. So next class onward, we are going to talk about a testing of our VLSI system that is all.

Thank you.