Embedded Systems – Design Verification and Test

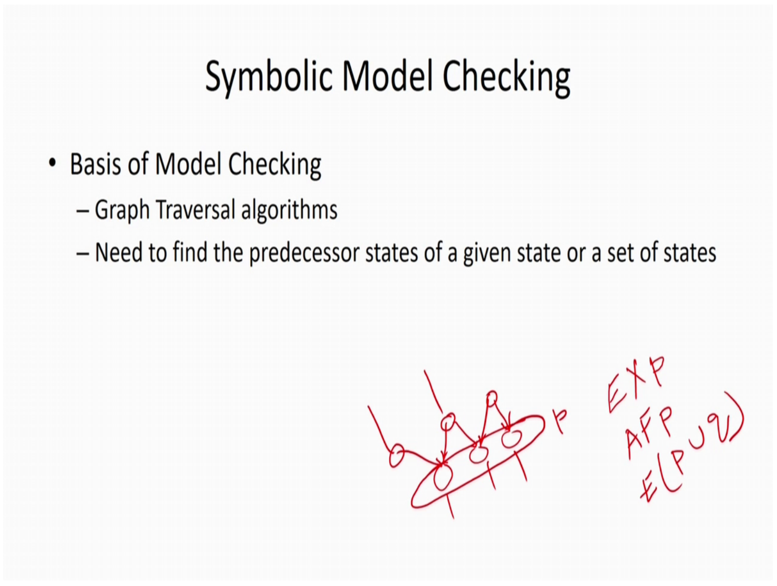
Dr. Santosh Biswas
Prof. Jatindra Kumar Deka
Dr. Arnab Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture – 25
Symbolic Model Checking

Hello everybody, Welcome back to the online course on Embedded System Design Verification and Test ok. In last class we have discussed how to represent and transition system with the help of an ordered binary decision diagram ok. Now, today we are going to see about a symbolic model checking. We already have discussed about the model checking algorithm.

In symbolic model checking algorithm, we are going to implement the same algorithm. But we are going to use data structure BDD to represent a transition system as well as set of states. Now, we will see how we are going to look for the correctness of a given CTL property in a BDD representation of the transient system ok.

(Refer Slide Time: 01:23)



Symbolic Model Checking

- Basis of Model Checking
  - Graph Traversal algorithms
  - Need to find the predecessor states of a given state or a set of states

p
EXp
AFp
E(pUq)

Already we know that, for model checking we should have a model which is some sort of your finite state machine. And we are having a CTL property, and one we are going to look for the correctness of this particular CTL property in a given model. Then the model checking algorithm returns as the set of states where the given CTL formula is true ok. Now already we have seen them; ROBDD can be used to represent the transition system ok. Because we are dealing with the transient system which is a finite state machine with additional property which is basically says about the leveling function. And we know how to represent all those things. So, this transition system can be represented with the help of ROBDD.

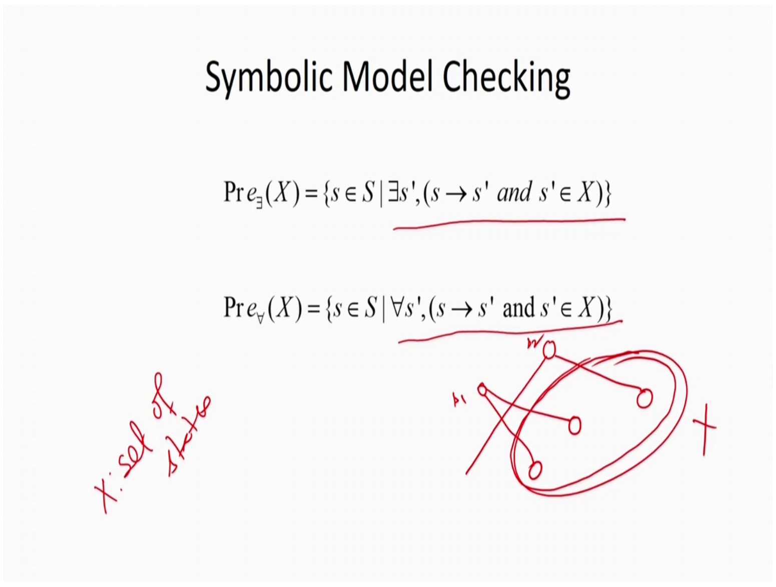
Similarly, if we are going to consider a set of states of that particular transition system then that set of states can be represented with the help of an ROBDD Reduced Ordered Binary Decision Diagram. So, these are the two bases that we have in our hand. Now we are going to use these two things to implement the model checking algorithm so which will be known as symbolic model checking.

So, what is a model checking algorithm? We know that it is a graph traversal algorithm and if you look into the operators. So, what about operators we are going to say if that here exist X say EXp it holds there exists a part EXp holds or in all path in future p holds, or maybe there exists a path, where p remains true until q becomes q maybe these are the three basic operators we can consider. And if we are having algorithm for these three methods then other methods can be expressed in terms of these three operators.

Now, when we are looking for the implementation of these three operator it is nothing but a, graph traversal algorithms. And what is the task of these particular graph traversal algorithms? Such we need to find out the Predecessor states of a given set of states or a given set ok. So, if I am having some states like that, then in my transition system, I may have some scenario. So, if say in this particular state say p is true. Then they are exist a path in next step p is true, then we have to find out what are the predecessor of those particulars states or these set of states.

So, this is the basic notions that you have to find out the predecessors state of a given state or a given set of states. So, this is next operator for future and until operator we have to do these job repetitively until and unless we are going to get a fixed point. That means, there is no more additional states in that particular set of state at that part we terminate ok, we terminates or algorithm. So, the basic crux is to find out the predecessors.

(Refer Slide Time: 04:56)


X: set of states
X

Symbolic Model Checking

$$\mathrm{Pre}_{\exists}(X) = \{s \in S \mid \exists s', (s \rightarrow s' \text{ and } s' \in X)\}$$

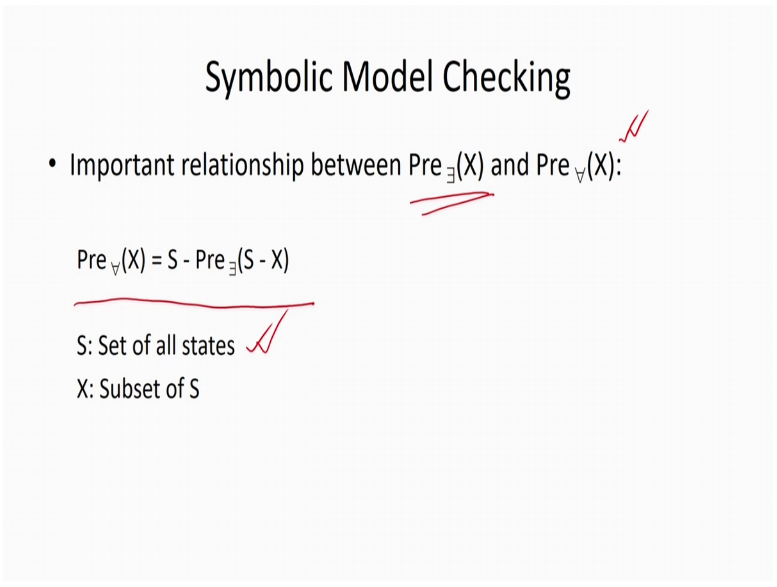
$$\mathrm{Pre}_{\forall}(X) = \{s \in S \mid \forall s', (s \rightarrow s' \text{ and } s' \in X)\}$$

Now, already we have seen that how to find out the predecessor states. So, we are having or we define two operators one is your Pre they are exist and Pre for all. So, Pre they are exist X, where X is a set of states. So, Pre they are exist says that at least, we are going to look for those particular states at least one transition is coming to the state of states; or committee a particular states which is a member of this particular state X ok. Or for all X; that means, you are going to look for those particular states where all the transition will lead to a next state which a member of this particular subset X ok.

So, if this is the scenario. Then for their exist we are going to get both the state s 1 and s 2, but for all we are going to get only s 1, because all the transition are coming to a states which is a member of this particular X. Now our job is to if I am having this particular set of states. Now I have to find out what are a predecessor of this particular state satisfying those particular properties for their existence it should satisfy this property and for all it should satisfy this particular property.

So, our job is to find out this particular set of states which are predecessor of this particular set of states. Now if we are representing our transition system with BDDS. Now we have to see how you are going to get those particular predecessor states. So, we are going to look for those particular scenarios. Again we have seen one important relationship between Pre they are exist X and Pre for all X.

(Refer Slide Time: 06:57)



Symbolic Model Checking

- Important relationship between $Pre_{\exists}(X)$ and $Pre_{\forall}(X)$:

$Pre_{\forall}(X) = S - Pre_{\exists}(S - X)$

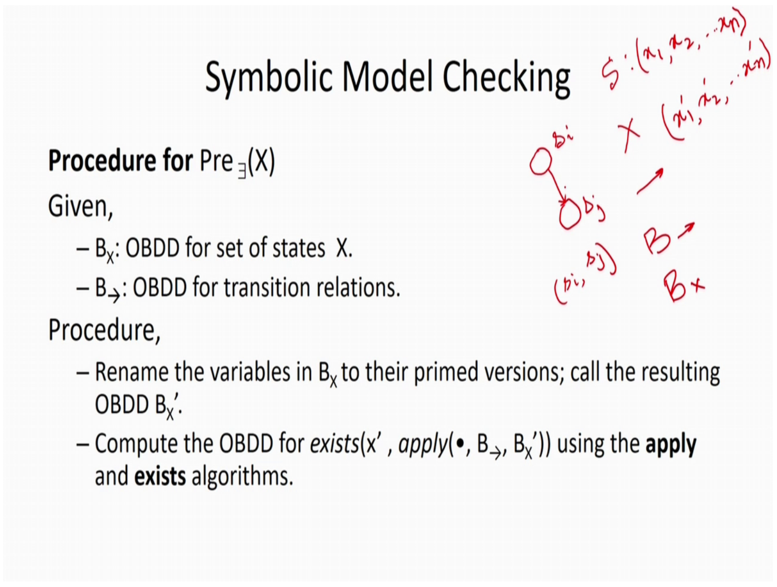
S: Set of all states

X: Subset of S

So, this is the relationship we have seen the Pre for all for all X is equal to S minus Pre they are exist S minus X. Where S is the set of all states, in a transition system we are having several state; we are going to consider all the states and we said this is the set S and X is a subset of S. Then from that we are going to find out what are the predecessor of those particular states which belong to X.

So, if we are having a method to find out to Pre they are exist X. Then we can use this method to find out Pre for all X and we will use this particular relationship. Now we are going to see how we are going to find out or have a method for Pre they are exist X ok.

(Refer Slide Time: 07:47)



Symbolic Model Checking

Procedure for Pre$_\exists$(X)

Given,

- B_X: OBDD for set of states X.
- $B_\rightarrow$: OBDD for transition relations.

Procedure,

- Rename the variables in B_X to their primed versions; call the resulting OBDD B_X'.
- Compute the OBDD for *exists*(x' , *apply*(•, $B_\rightarrow$, B_X')) using the **apply** and **exists** algorithms.

$S : (x_1, x_2, \dots x_n)$
$X \; (x_1', x_2', \dots x_n')$
s_i
s_j
(s_i, s_j)
$B_\rightarrow$
B_X

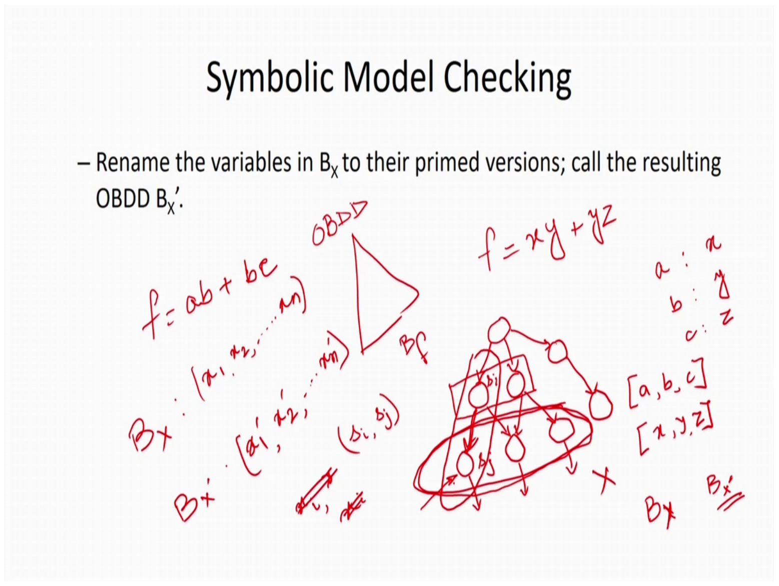
Now, this is simple method that we are going to say what is the procedure for Pre they are exist X, we are going to have the set of state X which is a subset of given set S ok. And we are having a transition relation ok.

Now, we are going to represent the transition relation with the help of BDD and we said this is the B arrow. So, B arrow is the BDD representation of the given transition system. And B X is the BDD representation of the set of states X. Now in the procedure we are having two step, in the first step we are saying that rename the variable in X to their prime version called a and call the resulting BDDs as B X prime. So, basically what will happen? To represent all the states so we are taking help of some variables say this is the variable set x 1, x 2, x n. If we are having total 2 to the power x n, 2 to the power n states then we need that many variable.

Now if I am having a transition from one particular state say s i to s j, then you are representing this transition like the order pair s i s j. So, s i will be represented by the state variables similarly s j will be represented by the state variable. To represent the next step we use another set of variable which are nothing, but the prime version of those particular variable; that means, x 1 prime x 2 prime x n prime.

So, this transition relation will be represented by x 1 to x n, and x 1 prime to x n. So, these are a variable that will be involved while we are going to represent this particular transition, like all transition will be represented by distance. Now B X is the OBDD representation or the particular state or a state of states ok. To, what will happen? We will just rename the variables of this particular BDDs.

(Refer Slide Time: 10:11)



Symbolic Model Checking

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

f = ab + bc
OBDD
Bf
f = xy + yz
a : x
b : y
c : z
[a, b, c]
[x, y, z]
Bx : (x1, x2, ..., xn)
Bx' : (x1', x2', ..., xn')
(si, sj)
si
sj
X
Bx
Bx'

So, what is the effect? You just say if I am going to have the scenario, like that if I am making a Boolean function f is equal to say ab plus say bc. And if I write the same expression, f is equal to say xy plus yz. Then both these Boolean expressions are going to represent the same function. Because what will happen? Here in one expression I am writing x, but this x variable is represented by x in the other one b it is renamed by y and c it is renamed by z.

So, which assume whatever we are having these variables are nothing but a, some sort of placeholder which can take values that are 0s and 1. So, variables can be different, but the functional values will be same. So, we can have similar scenario. So, this function can be represented with the help of an BDD or in particular OBDD.

So, we are going to get an representation which is basically B f. Then for this function also we are going to have the same Bf provided, we are using the same variable ordering; the variable ordering should be a b c for the first equation and x y z for the second equation. Because we are having this particular way mapping.

So, if we are having the same variable ordering. Basically with respect to their corresponding mapping we are going to get the same BDD. So, now, what happens we are having B x which is a BDD representation of the state of states. And it involves the variable say x 1, x 2 to x n. Now we can rename those particular variable ok, but still our BDD representation will remain same ok. That means, I am going to say that x 1 is replaced by x 1 prime, x 2 is replaced by x 2 prime, similarly x n will be replaced by x n prime. So, we are going to have the same BDD.

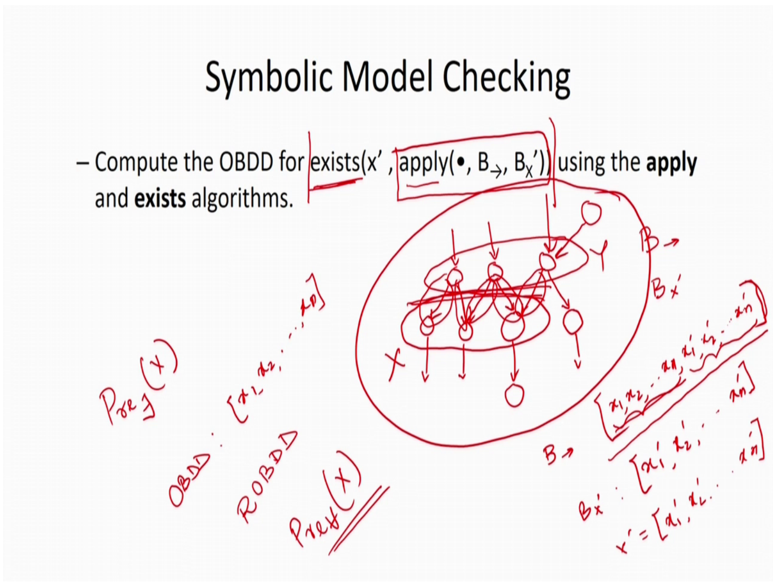
So, in that particular case what will happen? I say this is the BDD B X prime ok. Now consider a some small transition diagram or maybe say one finite state machine. So, if I am going to consider this particular set of state X ok. This states will be represented by variable x 1, x 2 and x n ok. Now this BDD whatever BDD I am getting over here I am going to say this is B X.

Now, I simply rename those particular variable by its prime version and we are going to get the BDD B X prime ok. That means, if we are having this thing that same BDD we are representing this particular state of state. But we are taking the next state variables, and now we want to find out the predecessor of those particular set of states ok. Then what will happen, what is the predecessor I must get? These two things ok. Now what is the relationship having those transition this is basically s i to s j, if this is say s i and s j.

Now, what is why we are doing that pre damming? Now you just see these transition i have to concept that on these transition to find out the predecessor of this particular state s j. So, this transition is represented by present state variable and next state variable. So, that is why we are saying we are renaming the variable of this particular state it is by next state variable and. Now the other state s j now we will map to this particular B X we will see how we are going to map these things. Basically we are going to look for this particular transition and going to have a predecessor of this thing. So, since we are concentrating on these things; that means, this transition has to be represented by its temporary variables.

So, in this particular case these state variables are basically now prime version and this state variable is my predecessor version ok. Now with the help of this renaming we try to find out does predecessor state.

(Refer Slide Time: 15:10)



Symbolic Model Checking

– Compute the OBDD for exists(x′ , apply(•, $B_\to$, B_X′)) using the **apply** and **exists** algorithms.

Y
X
$B_\to$
$B_{X'}$
$[x_1, x_2, \ldots x_n, x_1', x_2', \ldots x_n']$
$B_\to$
$B_{X'}$: $[x_1', x_2', \ldots x_n']$
$X' = [x_1', x_2' \ldots$
Pre∃(X)
OBDD : $[x_1, x_2, \ldots x_n]$
ROBDD
Pre∀(X)

Now, how we are going to do it? Now we are going to say that compute the OBDD for exist X prime apply dot B arrow the BDDs of the transition system, and B X prime using the apply algorithm and exist algorithm. So, when we are using this particular apply algorithm we are using two BDDs, BDDS for transition system; and BDDs for the set of states ok.

Now when we are going to apply this particular apply algorithm or use this particular apply algorithm then would B X prime or B arrow must have that compatible variable ordering. That means, they are going to use the same variable ordering. So, for B arrow I may think that it is the variable ordering x 1, x 2, x n, x 1 prime, x 2 prime, x n prime may be this is the variable ordering that I may have or B arrow. Then the variable ordering for B X prime will be your x 1 prime, x 2 prime, x n prime ok. Now they are having the compatible variable ordering, but B X prime is independent of those particular variables x 1 to x n.

So, this is the scenario now we are having. Now we are going to use the apply algorithm, and see what is the result that we are going to get. Now consider such type of scenario. And say we are considering that this is the state X, X is the set of states ok. Now we are using that particular B X prime to represent this particular set of states. And B prime is the, and B arrow is the representation of this particular enters that transition graph.

Now, when we use this particular algorithm apply. Then what will happen you just see, it is going to have an product of this interstate transition system, and this particular B X prime. Now while after applying this thing it is going to give me an BDD, that BDD is going to involve all the variables x 1 to x n and x 1 prime to x n ok. This is going to have these things, and basically if you see what we are going to get in this particular case; we are going to get basically those particular transitions. Because we are going to look for there exist a states ok.

So, basically it is returning this particular transitions ok. Now, all those transition are having variable from x 1 to x n and x 1 prime to x n prime. So, I am getting an OBDDs, which are having the ordering x 1 to x n and x 1 prime to x n, which basically returning us all those particular transition ok. Now, from those particular transition our objective is to find out the pressure states, where from this transition is originating. So, for those particular transition that presence that is represented by the variable x 1 to x n. And the next state is represented by x 1 bar prime to x n prime. Now we are going to use this particular exist algorithm on this particular x prime. Now what is this x prime? It is nothing, but the set of variables x 1 prime, x 2 prime, x n prime.

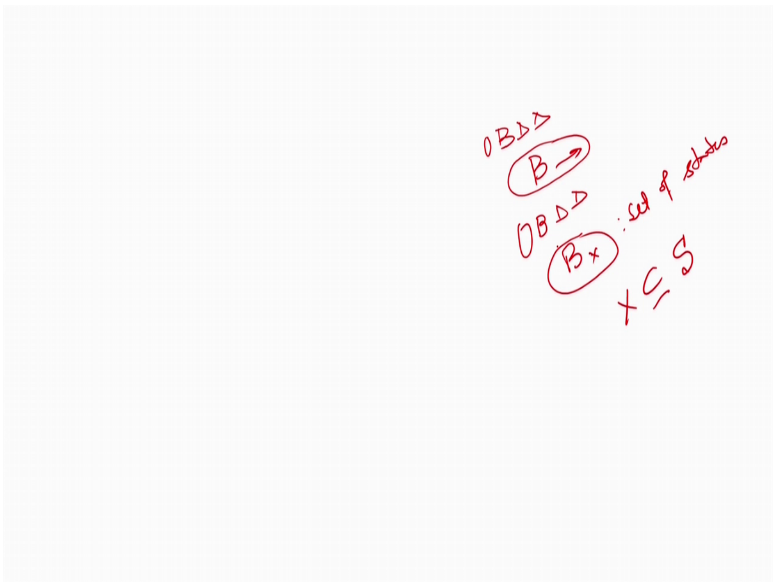
That means we are going to apply this particular exist on the resultant BDD on all the variable of x prime; that means, you are trying to relax the constraint on those particular variable x prime. So, when we are going to relax the constraint on the x prime then the resultant BDD is going to have the variable only x 1 to x n ok. Now, what we are considering? We are considering all those particular transition. And now we are relaxing on the next state variable, now we are going to get an BDD which have only the present state variable; that means, the resultant BDD is going to represent this particular set of states Y ok.

So, this is the inherent meaning of this particular operation. And this particular Pre there exist X method ok. Now from here what we have seen that, we can find out the predecessor states or by given set of state using this particular Pre they are exist X ok. Now whatever BDDs we are getting for this particular set of states Y. It is an OBDD, and it is going to have the same variable ordering the variable ordering will be your x 1, x 2 to x n ok. This is the way that we can look into it, now the resultant BDD may not be a reduced on then we can use those particular reduction techniques. That we know we are having only three reduction technique we can use those particular three reduction technique to get the ROBDD out of it ok.

Now, what we have seen, that if we are giving the transition system or representing the entire transition system with the help of an OBDDs, you are considering a set of state of that particular transition system. And representing those particular state of state with another OBDD, but the variable ordering of both the OBDDs must be computable. That means, they must have the same variable ordering then after application of these two operation apply an exist we are going to get an OBDD which indicates the predecessor states of the given set of states ok.

So, now we are having a method to find out the predecessor state of a given set of states which basically says that Pre there exist; that means, there exist at least one transition which are coming to this particular set of states. Now, we know the relationship between Pre for all X and Pre there exist X. So, now, we can use those particular relationships to get the Pre for all X also ok. Now we are having a method how to find out the predecessor state of a given set of state ok, already I have explained a method with example.

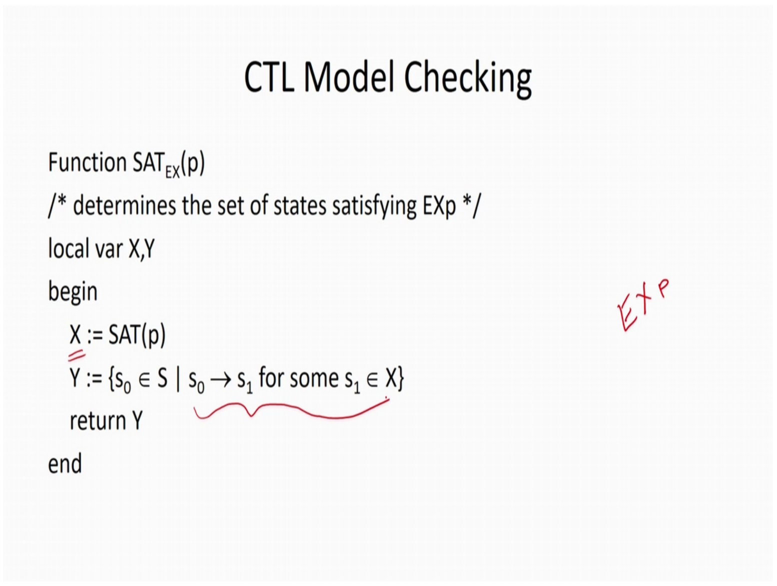
(Refer Slide Time: 23:07)



So, we are having an BDDs or basically say OBDD, B arrow for a state transition system we are having an OBDD, B X for set of states ok. So, x is a subset of the entire states S.

Then with the help of these two BDDs now we can find out the predecessor states of this particular set of states.

(Refer Slide Time: 23:40)



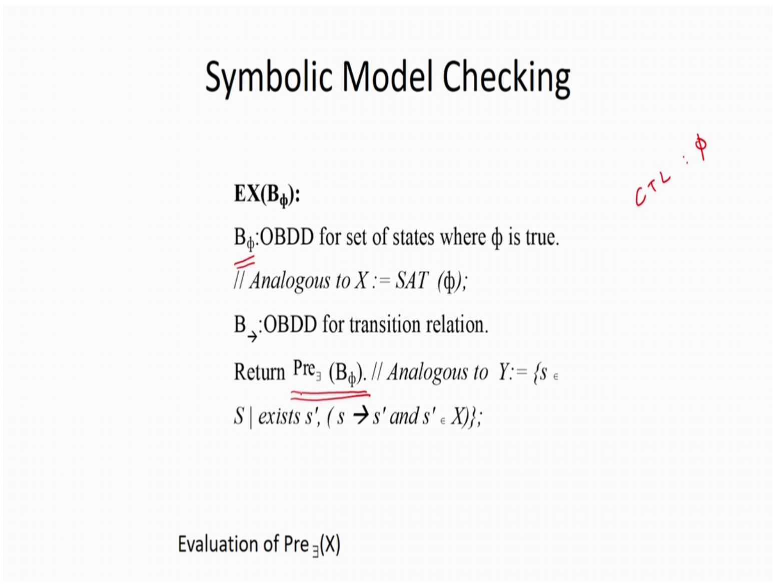
CTL Model Checking

```
Function $SAT_{EX}(p)$
/* determines the set of states satisfying EXp */
local var X,Y
begin
  X := SAT(p)
  Y := $\{s_0 \in S \mid s_0 \to s_1$ for some $s_1 \in X\}$
  return Y
end
```

Now, just see we are going to look for now the model checking algorithm. And see how we are going to have a symbolic implementation, and symbolic means we are going to represent or going to use OBDDs to the represent the entire systems and the set of states, and it says that we are not enamoring the set of state, but symbolically we are representing with the help of BDD. That is why the term symbolic is coming into picture and it says symbolic model checking.

Now what is the CTL model checking algorithm simply look into it, for operator EXp there exists a part next state p holds. So, we know the matt haughty it basically says that first we are going to take all the set of states where p is true. Then we are going to collects those particular states as 0 belongs to X, such that it satisfies these particular properties from s 0, there must be one transition to s 1 where s 1 belongs to X ok. Already we have seen this particular algorithm and finally, we written one. So, how we are going to represent we are going to have an implementation for this particular method symbolically.

(Refer Slide Time: 24:55)



Symbolic Model Checking

EX(B_ϕ):

B_ϕ:OBDD for set of states where ϕ is true.

// Analogous to X := SAT (ϕ);

$B_\rightarrow$:OBDD for transition relation.

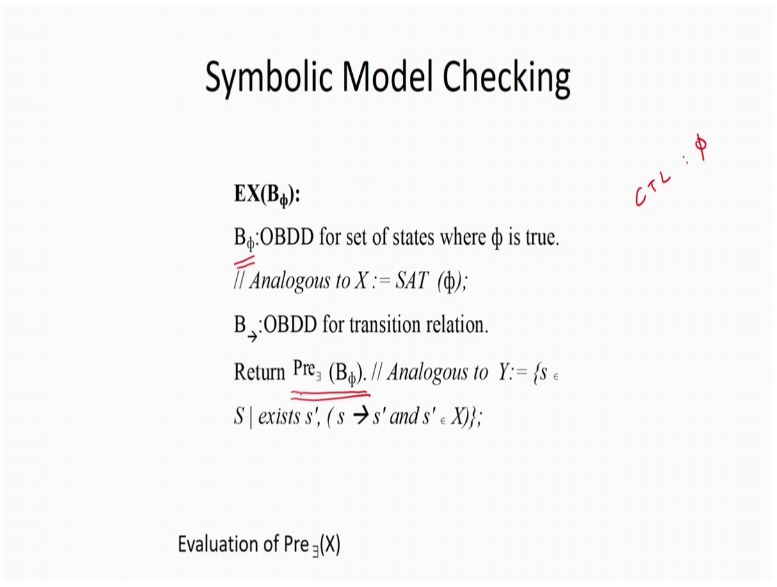
Return $Pre_\exists$ (B_ϕ). *// Analogous to Y:= {s ∈ S | exists s', (s → s' and s' ∈ X)};*

Evaluation of $Pre_\exists(X)$

So, this is the symbolic representation or maybe symbolic model checking algorithm, for there exist a part in next state some formula phi 2. So, in that particular case what will happen we are going to take the BDD B phi. It says that set of states where phi is true ok. Now we are having a CTL formula phi, now we know the set of state where phi is true. So, we are going to collect those particular set of states and going to have a BDD representation I made. And I said this is your b phi. So, this is analogous to X equal to set of phi. The way we are saying in this particular statement ok. So, this is analogous to this particular statement of the model checking algorithm. Now, B arrow is the OBDD representation of the transition relation or entire transition system. And now, B phi and B arrow must that, compatible variable ordering.

Now after that we are just using this particular operator Pre there exists B phi. B phi is the BDD representation of the states where the formula phi is true ok. So, it is analogous to like that we are going to loop for the predecessor states s such that from S we are having a transition to s dash and s dash belongs to X. This is the predecessor operator basically does. So, finally, we are going to return this particular thing, and what it is going to return? It will return a BDD, that BDD is going to basically talk about or going to represent a state of states which is having a transition to this particular set of state X or basically these are the state of states where that formula EX phi is true.

(Refer Slide Time: 26:51)



Symbolic Model Checking

EX(B_ϕ):

B_ϕ:OBDD for set of states where ϕ is true.

// Analogous to $X := SAT(\phi)$;

$B_\rightarrow$:OBDD for transition relation.

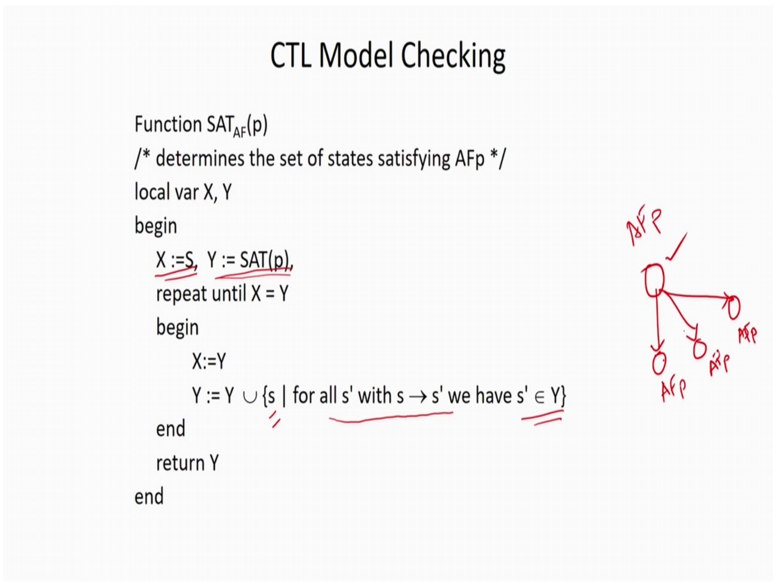
Return $Pre_\exists(B_\phi)$. *// Analogous to $Y := \{s \in S \mid \text{exists } s', (s \rightarrow s' \text{ and } s' \in X)\}$;*

Evaluation of $Pre_\exists(X)$

So, basically now we can think something like that, we are talking about EX phi. So, consider a small transition system, now just say these are the states who have phi is 2; that means, I am going to get this particular set of states. And I say this is basically B phi. This is the starting point and I am going to have the BDD representation of this particular entire state space, which I said this is your B arrow. Now we apply this particular model checking algorithm, then what will happen? It is going to give me those particular states where the in next step phi is true.

So, in this particular case it is going to return me those particular state maybe this is s 0 s 1 and s 2 ok. Because in this particular state the next state five is old, for here also at least I am going to have one state. So, after performing this operation it is going to return me this particular set of state s 0, s 1, s 2 where this particular formula EX phi is 2. So, this is the way I can look into it, now I am having the implementation of this particular EX operator there exist a part in next state phi holes where we are using the BDD representation only, BDD representation of the transition system and BDD representation of the set of state.

(Refer Slide Time: 28:58)



CTL Model Checking

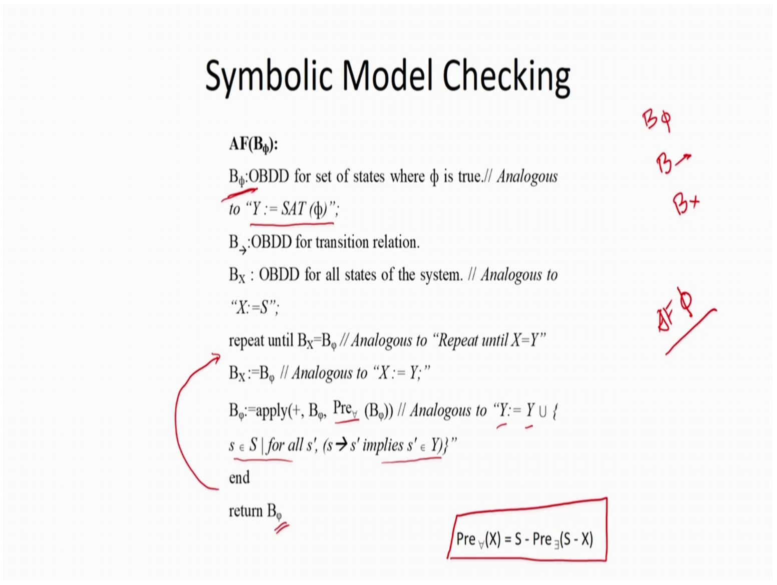
```
Function SAT_AF(p)
/* determines the set of states satisfying AFp */
local var X, Y
begin
  X := S, Y := SAT(p),
  repeat until X = Y
  begin
     X:=Y
     Y := Y ∪ {s | for all s' with s → s' we have s' ∈ Y}
  end
  return Y
end
```

AFp
AFp
AFp
AFp

Now, similarly the another operator is your A F p. There exists a part oh sorry A F p says that in all part in future, we are going to some state where p is true. And this is the algorithm that we are having first X is equal to S; that means, I am going to take the entire state space and Y is equal to SAT of p, where I am going to take the set of state where this particular atomic sorry CTL formula p is true. This is the satisfy some relation. So, I am going to considered all the states where p is true. Now repeat until X equal to Y, we are going to repeat it like that when X become equal to Y and initially I am going to mark X is equal to Y.

That means wherever that p is true in the particular state A F p is also true. Because this is as per the motion of our time future includes the presence in area also. So, this is the initial state where that A F p is true. Then I am going to collect such type of state and going to append with Y or union with Y, what those particulars must have this particular property for all s dash, we have a with the transition s to s dash, we must have all as this must belongs to Y; that means, if here A F p is true. Then the assigning A F p will be true over here also because all the transition is coming to a state where A F p is true. So, this is the scenario now we have these things will be again implemented with the help of symbolic model checking algorithm where we are going to use the BDDs.

(Refer Slide Time: 30:51)



Symbolic Model Checking

```
AF($B_\phi$):
$B_\phi$:OBDD for set of states where $\phi$ is true.// Analogous to "Y := SAT ($\phi$)";
$B_\rightarrow$:OBDD for transition relation.
$B_X$ : OBDD for all states of the system. // Analogous to "X:=S";
repeat until $B_X$=$B_\varphi$ // Analogous to "Repeat until X=Y"
    $B_X$:=$B_\varphi$ // Analogous to "X := Y;"
    $B_\varphi$:=apply(+, $B_\varphi$, $Pre_\forall$ ($B_\varphi$)) // Analogous to "Y:= Y ∪ { s ∈ S | for all s', (s → s' implies s' ∈ Y)}"
end
return $B_\varphi$
```

$\boxed{Pre_\forall(X) = S - Pre_\exists(S - X)}$

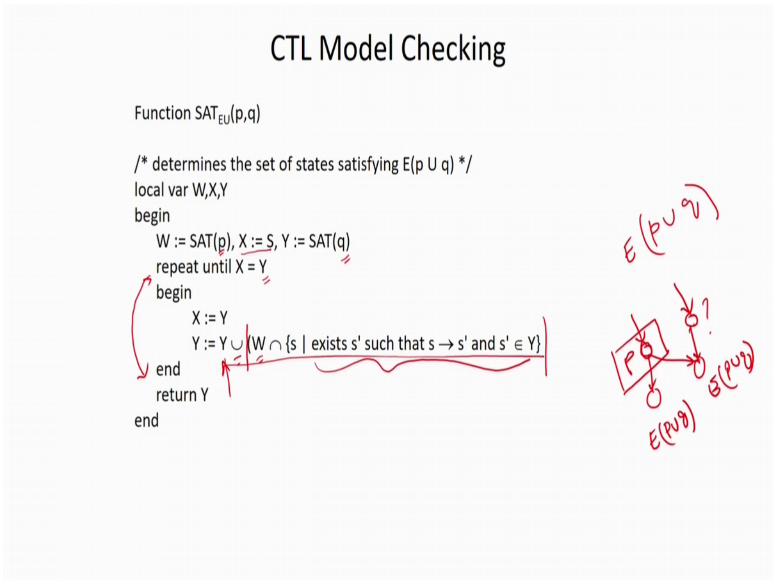
So, similar to your EX phi here I am going to say that B phi is an OBDD representation of the state of state where phi is true. This is analogous to Y is equal to certification of part. B arrow is the OBDD representation of the transition relation ok. So, one I am taking B phi these are the state of BDD representation of the set of state where phi is true, B arrow is the BDD representation of the transition system. And B X is the BDD representation of the all states of the system, say one is your transition relation and B X is the all states, this is the BDD representation. So, this is analogous to your X equal to S. So, these are the three things I am having X equal to X and Y equal to set of p.

Now what we are doing repeat until B X equal to B phi; that means, BDD a presentation of the set of state where phi is true will be equal to the previous iteration. Basically, now here first step we are going to assign B phi is equal to BX; that means, we are returning a set of state where phi is true. Because we are going to appends states to this particular set of state Y; that means, you are going to appends state to this particular state or to this particular B phi. Now what will happen? Now B phi will be now we are going to modify it. How we are going to modify it? Apply plus B phi and Pre for all B phi ok. So, this is analogous to that we are appending Y with Y union and all such type of states which satisfies this particular property.

So, we are using now Pre for all, and we know that Pre for all can be observed by this particular relationship Pre for all X equal to S minus Pre they are exists S minus X. So, we are having a matter to this thing. So, this is plus is going to have the union. So, we have a use this particular apply method or apply algorithm to get the union of these things. And we are going to repeat it (until and unless step that Previous states or set of state this is equal to the set of states that we are getting after this iteration.

So, once these two BDDs become equal; that means, we are not adding any more new states to it and we terminated and finally we return this particular B phi. Now B phi is the BDD representation of the state of states where this particular AF phi is true. We are looking for the formula a f phi. So, we are getting an ordered BDD representation or the state of states where this particular formula is true. Now after that we can use those particularly reduction technique to get the ROBDD representation of those particular state of states. Now this method is giving us the returns the state of state where this particular formula is true and it gives me the BDD representation of those particular set of states ok.

(Refer Slide Time: 34:17)



CTL Model Checking

```
Function SAT$_{EU}$(p,q)

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

E(p∪q)
p
?
E(p∪q)
E(p∪q)

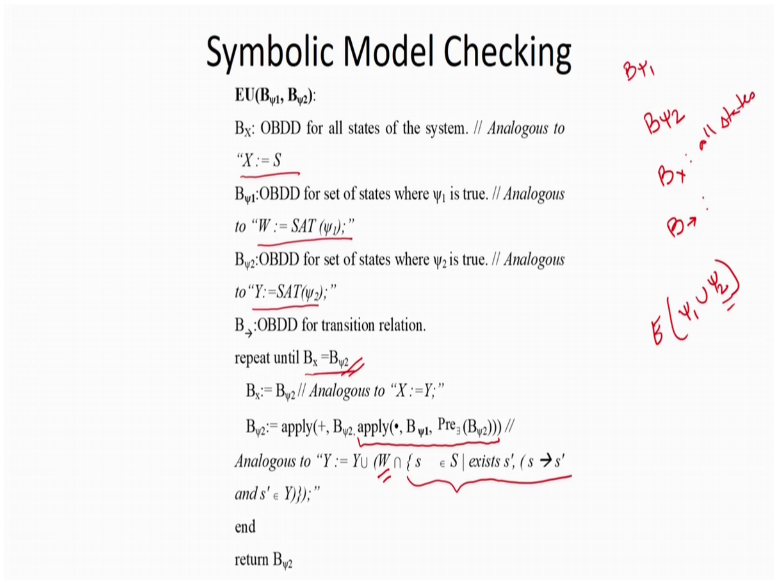
Now we are having another porter called e until ok. So, p until q or there exists a part p until q. So, this is the method, now what we are doing? Going to take the set of state where p is true, we are going to take the set of state where q is true. And we are going to look for entire states all the states of the system ok. Now we are repeating until X equal to Y then we are going to these things. Now in Y these are the set of state where q is true.

Now we know that where q is true, then E p until q is true according to our notion of time, where we says that presence or future includes the present behavior also.

So, wherever q is true and E p until q is true. So, that is why this is the initial state of state where E p until q is true. Now X equal to Y. Now we are retaining the previous set of states now here this is the operation that we generally perform already we have discussed in our model checking algorithm. So, Y union and what sort of state of states we are going to take? We are going to consider those particular state of state, we are going to collect those particular states which is going to satisfy this particular property; that means, at least there is one transition to the set Y and after that intersecting with W; that means, where p is true.

So, we are going to look for such type of scenario. If say E p until q is true over here, E p until q is true over here and say it is p is true over here. Then when I am going to take the predecessor state? We are going to look for those particular Predecessor state where p is true; that means, this particular state will come into picture because this is the W union and what is W? Where the set of state where p is true so these predecessors will come, but this Predecessor will not come interface. So, this is the way we generally collect the set of state where p until q is true. And we repeat these things until and unless we cannot add any more new step.

(Refer Slide Time: 36:47)



Symbolic Model Checking

```
EU($B_{\psi1}$, $B_{\psi2}$):
$B_X$: OBDD for all states of the system. // Analogous to "$X := S$
$B_{\psi1}$:OBDD for set of states where $\psi_1$ is true. // Analogous to "$W := SAT(\psi_1)$;"
$B_{\psi2}$:OBDD for set of states where $\psi_2$ is true. // Analogous to "$Y := SAT(\psi_2)$;"
$B_\rightarrow$:OBDD for transition relation.
repeat until $B_X = B_{\psi2}$
    $B_X := B_{\psi2}$ // Analogous to "$X := Y$;"
    $B_{\psi2} :=$ apply(+, $B_{\psi2}$, apply(•, $B_{\psi1}$, $Pre_\exists(B_{\psi2})$))) // Analogous to "$Y := Y \cup (W \cap \{s \in S \mid$ exists $s'$, $(s \rightarrow s'$ and $s' \in Y)\})$;"
end
return $B_{\psi2}$
```

So, this is the represent symbolic model checking implementation, where we are using the ordered BDD. So, this is B psi one is the BDD representation of the state of state where psi 1 is true. And B psi 2 is the set of state where the CTL formula psi 2 is true. So, these are the two BDDs we are having then, we are going to take another BDD B X which is going to BDD representation of all states ok. And B arrow is the BDD representation of the transition system.

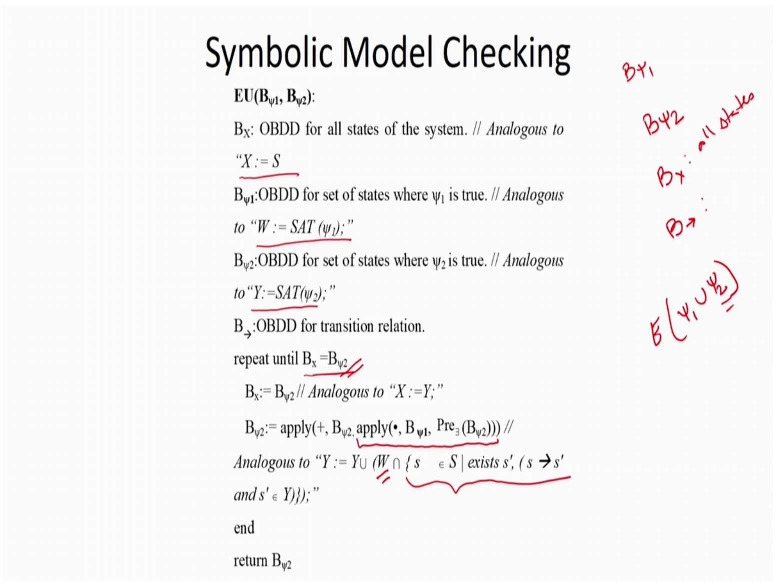
Now all those 4 BDDs must have compatible variable ordering then only we can perform the operation. So, it says that B x is the OBDD representation of the system this is analogous to X equal to S if you compare that or model checking algorithm. This is B phi 1 is the OBDD representation of the set of state where psi 1 is true. This is analogous to W into set of psi 1, B psi 2 is the BDD or OBDD representation of the set of states whereas, psi two is true. So, this is analogous to Y is equal to set of psi 2 and B arrow is the OBDD representation of the transition direction. So, these are the four BDDs we are using.

Now, repeat until B X is equal to B psi which is analogous to your X equal to Y and here you say that initially wherever that psi 2 is true we are assigning to B X ok. So, again you just see that whenever we are going to have E. There exist a path psi 1 until psi 2 all the state where psi 2 is true. In those particular states we said that E psi 1 until psi 2 is true. That is why this is the initial set of state or BDD representation of the initial state of state where psi 2 is true. Now we are going to conduct more and more step. So, what is this? So, this is the scenario first we are using this particular apply dot to B psi 1 and these things. So, this is basically B psi 1 is the set of BDD representation of the state of state where psi 1 is true. This is equivalent to this W ok. Representation of the set of state W and this is the intersection of such type of states.

So, that is why for this intersection we are using this particular dot operators. So, B psi 1 Pre there exist B up psi 2 ok. Taking all the states like that previous example all the taking predecessor states where the E p until q is 2 and p must be true over l. So, this is W intersection of these things. So, this is dot operation is there and we are going to add those particular things to the state of state where already we have founded, E psi 1 until psi 2 is true this is the union. So, this union is again given by this particular plus operator.

So, we have applied or use this particular applied method twice to find out B psi and we repeat these things until and unless it is going to satisfy these particular property. Then finally, we are going to return B psi 2 where this particular formula E psi 1 until psi 2 is true. Now you just see this is the symbolic implementation or I can say that where we have used the BDD representation for the entire system. And a state of states to implements the operator E psi 1 until psi 2 ok.

(Refer Slide Time: 40:49)



Symbolic Model Checking

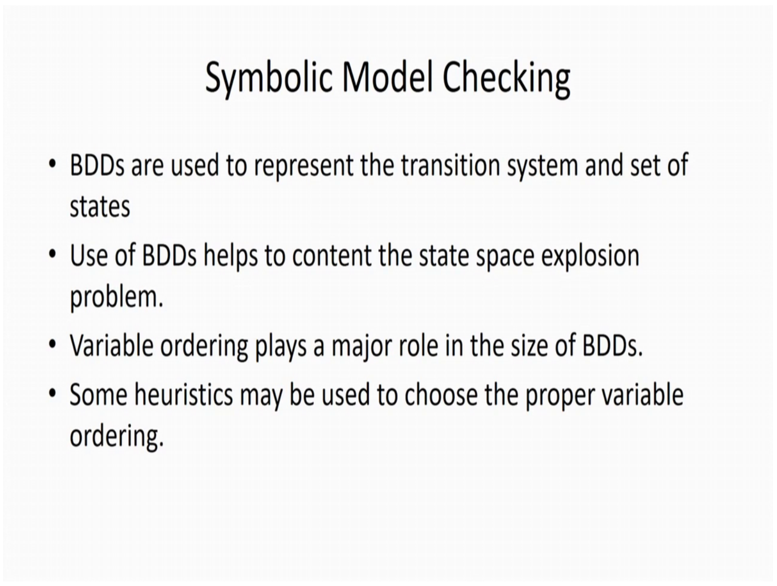
```
EU($B_{\psi1}$, $B_{\psi2}$):
$B_X$: OBDD for all states of the system. // Analogous to "$X := S$
$B_{\psi1}$:OBDD for set of states where $\psi_1$ is true. // Analogous to "$W := SAT(\psi_1)$;"
$B_{\psi2}$:OBDD for set of states where $\psi_2$ is true. // Analogous to "$Y:=SAT(\psi_2)$;"
$B_{\rightarrow}$:OBDD for transition relation.
repeat until $B_X = B_{\psi2}$
  $B_X := B_{\psi2}$ // Analogous to "$X := Y$;"
  $B_{\psi2}$:= apply(+, $B_{\psi2}$, apply(•, $B_{\psi1}$, $Pre_\exists(B_{\psi2})$)) // Analogous to "$Y := Y \cup (W \cap \{s \in S \mid$ exists $s'$, $(s \rightarrow s'$ and $s' \in Y)\})$;"
end
return $B_{\psi2}$
```

Now, what happens we have seen these things for EX p AF p and E p until q. And we know that, if we are having of implementation of these three operators. Then other operators can be converted or can be written with respect to these three operators.

So, now we have a method where we have use the BDDs to find out the state of state where a given CTL formula is true. Now when we are using OBDD's or say ROBDD's up to some extent we able to contain the states plus expression problem. Because in many a case it is observed that the OBDD representation of the entire system or maybe set of state coming to be a very small one if we can find out a proper variable ordering.

So, we try do we use defined heuristics initially to find out the appropriate variable ordering. And once you get the appropriate variable ordering then we use this particular variable ordering to represents to represent our entire state space as well as the set of states. So, when you use this particular symbolic method, symbolic model checking we can take care of many more states ok. It is at least getting better result with compared to the graph Bessel algorithm or graph traversal algorithm. So, that we have thus implementation of model checking algorithm with BDDs alright.

(Refer Slide Time: 42:32)



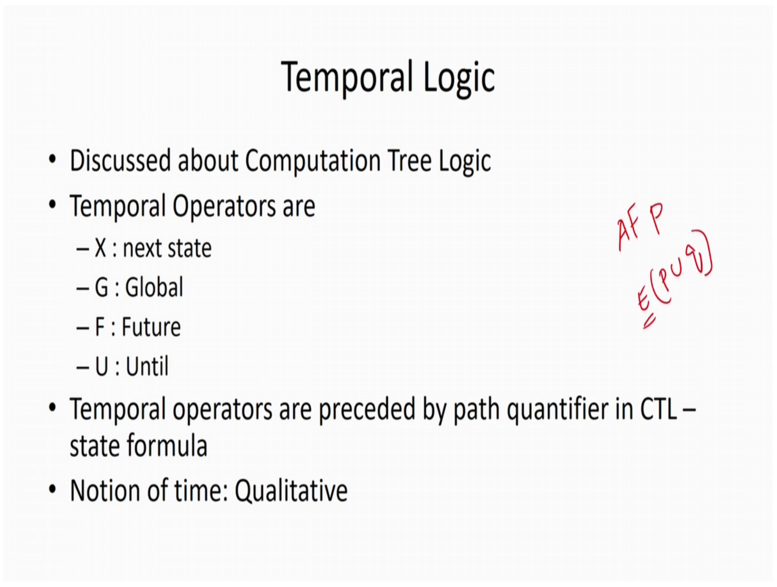
Symbolic Model Checking

- BDDs are used to represent the transition system and set of states
- Use of BDDs helps to content the state space explosion problem.
- Variable ordering plays a major role in the size of BDDs.
- Some heuristics may be used to choose the proper variable ordering.

Now, whatever this things so whatever we said this is the in nutshell I can say that BDDs are used to represent the transition system and set of states or in particular I can say it is the ordered BDD. Because we are using several BDDs and we are performing operation of several BDDs. So, they must have a computable variable ordering. So, we are using ordered BDDs to represent transition system and a set of state. Use of ordered BDD helps to contain the state space expression problem. Because already I have mentioned that in many a times or many a cases we are going to get a compact representation of those particular set of states and the transition relation, but main thing is that we have to find get an appropriate variable ordering.

So, here already I said that variable ordering plays a major role in the size of the BDDs. So, we use some heuristic to shows the proper variable ordering. So, these are the basic ones that we have to know, we have to keep in our mind while we are going to implement the model checking algorithm symbolic where; that means, we are going to use the OBDDs to implement our model checking algorithm.

(Refer Slide Time: 43:44)



Temporal Logic

- Discussed about Computation Tree Logic
- Temporal Operators are
  - X : next state
  - G : Global
  - F : Future
  - U : Until
- Temporal operators are preceded by path quantifier in CTL – state formula
- Notion of time: Qualitative

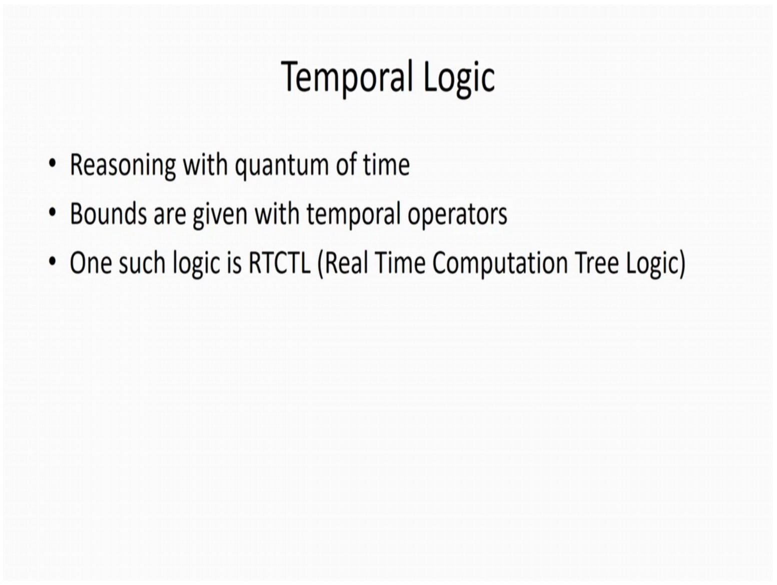
Now, along with that with model checking what we need we need a temporal logic ok, and in this discussion we are using a particular temporal logic which is called CTL computational tree logic. And what are the temporal operators we are having? Basically we are having the next state global user and until.

And for CTL all those temporal operator will be preceded by a part quantifier, there are two part quantifier a and e a is for all part, e is for there exist a apart. And when you proceeded in those particular temporal operator with part quantifier that all becomes a step formula; that means, we are specifying the properties of the system in a state. And CTL model checking basically taken care of the state formulas only, and whatever operators we have discussed till now that X G F and U here the, if you look into the notion of times these are qualitative illnesses, what we say? A F p whether in all path in future p holds; that means, we are going to look for all the transition from a particular state in such a way that somewhere in future or eventually we are going to get a state where p is true.

So, this is quantitative in nature we are saying that in future we have to get it ok. And similarly, if I say that E p until q what we are saying. That p must remain true eventually q becomes 2 in a particular state; in an particular execution or in a particular execution trace because it is taking for E part quantifying; that means, we are not concerned about all the part.

But at least in some path p must remain true until q becomes true. Again it is qualitative in nature we are not having any quantity we are not saying that in all path in future we did not 5 units of time p must hold true. So, that will be quantitative in nature.

(Refer Slide Time: 46:03)



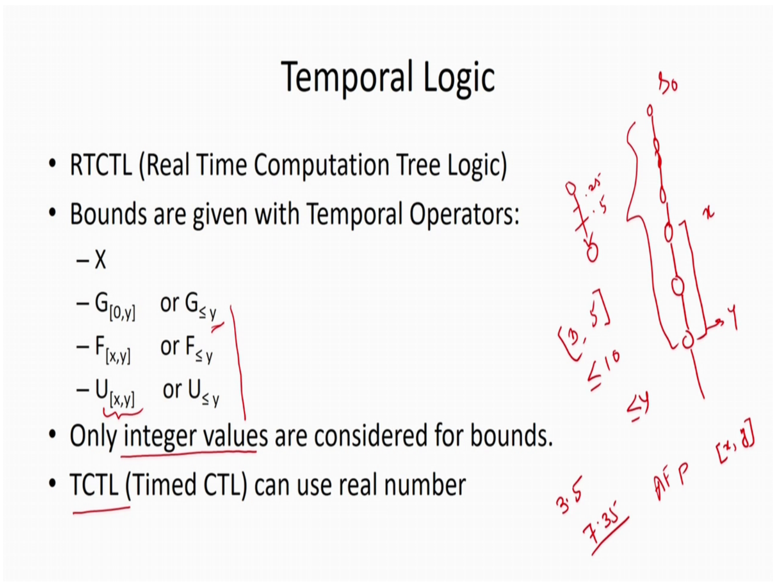
Temporal Logic

- Reasoning with quantum of time
- Bounds are given with temporal operators
- One such logic is RTCTL (Real Time Computation Tree Logic)

So, how to look for the quantity of time? So, quantitative in nature; that means, I can say that I am looking for some property to be to hold within say 5 unit of times or maybe in between 7 to 10 unit of times. So, these are the time notions. So, along with the temporal operators we can give the bounds also.

So, one such logic is developed we say on this logic is called RTCTLs this is Real Time Computation Tree Logic. So, one such logic is there RTCTL where we say that time will be specified in quantum ok; that means, it is quantity business or whatever algorithm we have discussed this is qualitative in nature. But now, we are going to specify the quantum of time. So, we are going to say that that time will be quantitative in nature, which is very much required in a real time system. Because always we want to have the response with a specific time. So, with the CTL operators or temporal operators we can give a bound to indicate the quantum of time. So, for that what will happen all those operators will be given as some bounds.

(Refer Slide Time: 47:19)



Temporal Logic

- RTCTL (Real Time Computation Tree Logic)
- Bounds are given with Temporal Operators:
  - X
  - $G_{[0,y]}$ or $G_{\leq y}$
  - $F_{[x,y]}$ or $F_{\leq y}$
  - $U_{[x,y]}$ or $U_{\leq y}$
- Only integer values are considered for bounds.
- TCTL (Timed CTL) can use real number

s_0
x
y
[3, 5]
≤10
≤y
3.5
7.35
AF p
[x, y]

So, X is basically next state, we do not have any problem if we are having a step in the next time it should get it, but for G globally whether we are going to look for all features or we are going to bound it to sometimes. So, it will be something like that at has looking into some bound Y. So, within this Y all state should satisfy this property or in some notion it says that from current time present state to that particular bound Y.

So, we can help the sub script bound 0 to Y or we can say that less than Y; that means, from this particular state through the time instance Y. So, similarly in future we can say that within this particular Y time unit that p must be true or that given property must be true or we can give a bound that in between X and Y it must be true. That means, if I look into transition part. Then if I say it must be less than equal to y and say this is the time instance y; that means, from this particular state somewhere p must be true.

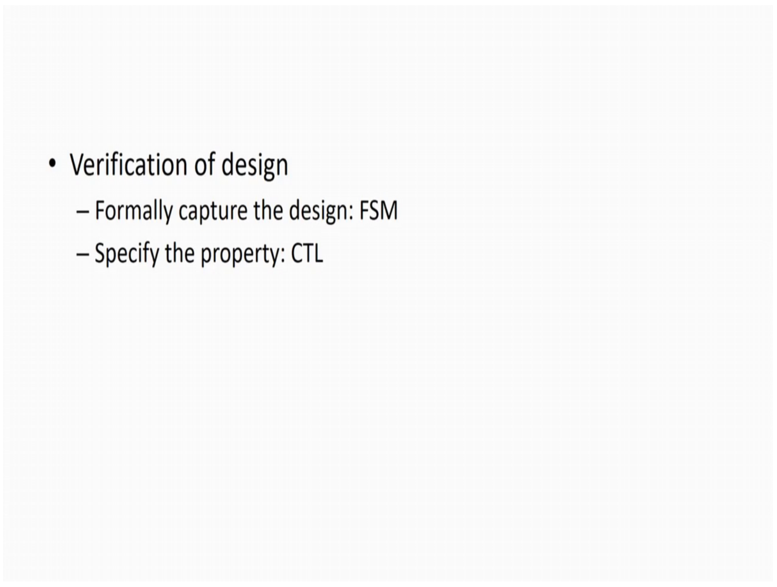
If I am going to look for say AF p or AF p or if I give a bound like that x and y then p must be true during within this particular bound. So, both are possible similarly in case of until also I can give a bound and you can say that less than equal to Y. So, we (Refer Time: 49:07) or maybe that total bound from that current state to thing things. Now, this is the way we can have the quantitative in nature how can implement the, or we can indicate the quantity. So, those algorithm whatever model checking algorithm we have discussed, those can be implemented to get take care of those particular bounds also you have in the implementation.

So, in RTCTLs real time computational tree logic competition tree logic we generally provide such type of bound. And we say that within some specific time my given properties would hold. So, such type of reasoning can also be done if we are going to use those particular bound or reuse the RTCTL. Now, what is the basic restriction or limitation of this particular RTCTL? Here the bounds basically takes integer values only, maybe you can say that within time period time stamp 3 point 3 and 5 or maybe within 10 time units.

So, it can take integer values only. But now people are looking for the real handle real time also. So, for that we are having one logical TCTL time CTL. So, in case of time CTL then what will happen we are trying to consider the real numbers also as a bound maybe you can give 3.5 or say 7.35 like that, but the system becomes more complex when we are going to handle real time. Because you can see that if I am having a transition maybe in quantum of time I can say this is in one time unit it can come to the next state ok.

So, now when I go for the real time then what will happen I consider within 0.5 may be 0.25 all those particular states will come on all those particular behavior you have to take care of it. So, this Y that we are going to have a time region and p, people have come up resources have come up with a method where we can handle the real numbers less time. But system is very complex ok. So, it is we are not going to discuss those particular methods or particular issues in this particular lecture just I am giving an idea that we can handle quantum of time, quantum of time may be integers or quantum of time may be your real numbers also. So, if you are interested then you can have a look into those particular research papers where people have introduced TCTL .

(Refer Slide Time: 52:00)



- Verification of design
  - Formally capture the design: FSM
  - Specify the property: CTL

Now, what we have discussed till now? There you see that we are going to design a system then we know the behavior, say when we are going to design a system we know the behavior of the system. How it is going to be a so; that means, it is that system is having different configuration that configuration is basically nothing, but the combination of those particular state variables. And we are having several transition from one state to the other state.

So, if a particular instance say if we are doing some activities then it should lead to some result. Most of you might have seen the coffee vending machine, if you use the coffee vending machine. Then what will happen? You go to the coffee vending machine and you press the button coffee then what will happen, it will take the premix of the coffee it will bring it out then it will mix with your milk and finally, it will give it to the coffee, readymade coffee in your cup that this place inside the outlet. In the same machine if you are going to press the button tea then what will happen it will take the premix of tea mix it with water and finally, give you the tea.

So, how we are going to control all those things? So, when I press coffee then what will happen? I have to open the part where I have kept the premix of the coffee. When I press the tea then it must open the fall, where we have kept the premix of the tea. Now you just see; that means, in one particular state the coffee fall of the coffee is on, but tea and other stocks are off. So, this is a particular configuration where I am going to make coffee. So, another configuration I am going to get where the pulp for tea is open, but the pulp others items are closed.

So, this is another configuration. And we are having transition from one to the other depending on the way we are going to use this particular coffee vending machine. So, this behavior we are going to captured in our design. And generally we are going to get a finite states and most of the cases we are going to use the finite state machines. And here in model checking what happens we said it is a crisp case structure because it is having two more criteria's need to be satisfied. One is that whatever transition relation we have it must be complete; that means form all step there should be an next state also.

And Secondly, we are having a leveling function. With the help of leveling function so we are going to say, what are the atomic proposition to in a particular state. So, this is formally we are capturing our design. Then we know that whenever we are going to design it it must satisfy some property. So, in that particular case those properties we are going to capture with the help of CTL. Because in this class we have discussed about the CTL model checking only, but you can look for some other verifier also where we can specify the property in some other way.

So, these are the two things because we know my system should behave in this particular way. So, that behavior we are going to captured with my CTL which is a property of this thing. Then we use the model checker to check whether this property holds in this particular model or not. So, if properties is true in my model then what will happen that model verifier is going to give say going to give me the answer yes.

So, if all the properties are true then what will happen I can go for the implementation of my design ok? Now if the some of the properties are not true. Then what will happen that model verifier will say that what about system you have designed it is not satisfying some of the properties, along with that it will give me one indication or you can give me a press to so that, if you follow this particular execution trace the given property is falls.

Then as a designer what I can do? I will consider on those parts of my design to rectify those particular errors and finally, again I am going to look for re verification. Once it is correct it is give me some confidence that whatever I have designed it is correct. Then I will go for the next phase of my design or next phase of for design phase where I am going to implement this particular design. So, this is the states that involve when we are going to design it. Now, when we are coming to the embedded system basically what we have seen? embedded system is nothing, but a combination of hardware and soft software; that means, we are going to design a system where both hardware and software's are involved; that means, some part will be implemented in a hardware and some part will be implemented as a software program.

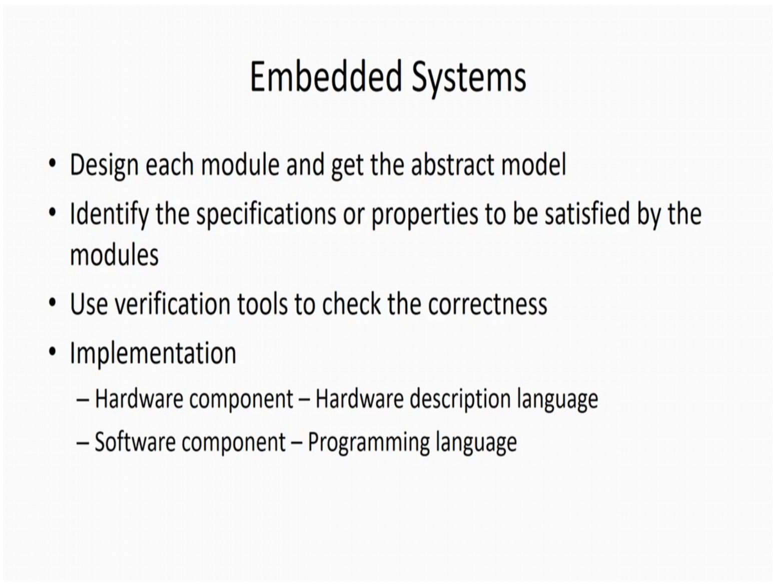
So, when I am going to design a system to fulfill some of my criteria some of my requirements. Then we are going to identify what are the different tags that we have in the system. What are different tags we need to perform? A nd for every task we are going to have the modules. When we say that these are a models that we need to implement to get the solution to get the final product, now after having those particular tasks or particular module now we have to identify what are a models that we are going to implement in hardware and what are a modules we are going to implement in software. If you are going to implement everything is hardware its fine, but why you are looking for both the option going to implement something in hardware and going to implement something here software just to meet all our requirement.

One is that we are having some per formant criteria's that my system should respond well behave within specific time. Which is basically their performance another criteria we are having that cost factor. It should remain within my budget it should not exceed my budget because if it exceed my budget then my product will be very costly.

So, for that now we are going to identify those modules where I the module is your performance in (Refer Time: 58:56) that means, you should walk in a passed away. So, for those basically we are going to select for hardware implementation. Because if you directly implement something in a hardware. Then it will walk faster our response will be very good. Now other component where it is not that effective that we can before time sometimes to get the responses, but we are going to cut the cost or reduce the cost then those component we will identify, and we said that for them we are going to have an software implementation; that means, at that software will be used some resources common resources to execute that particular software.

So, this is the way we can look into it; that means, you have to identify the hardware part and you kept to identify the software part.

(Refer Slide Time: 59:51)



Embedded Systems

- Design each module and get the abstract model
- Identify the specifications or properties to be satisfied by the modules
- Use verification tools to check the correctness
- Implementation
  - Hardware component – Hardware description language
  - Software component – Programming language

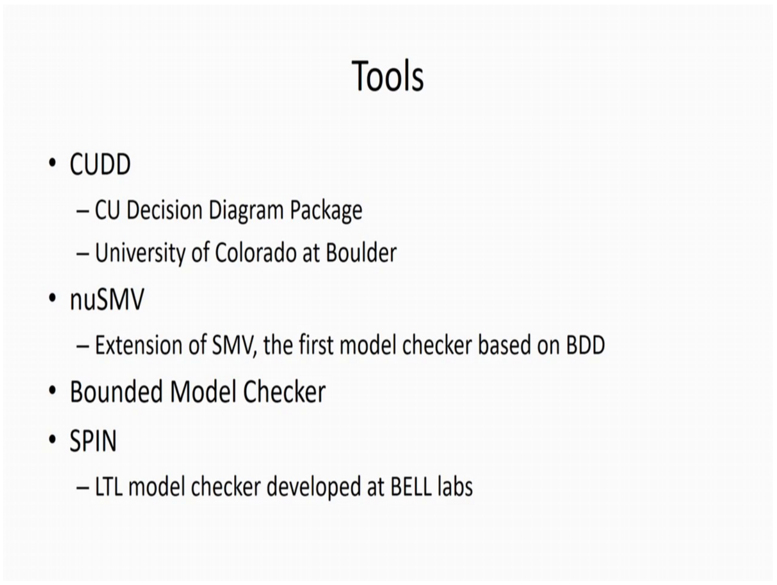
Now, what we are going to do? We are going to design its module the way already I have mentioned. And get the abstract model of those particular modules; that means, we are going to have a formal representation of those particular modules. And we said this may be some sort of your finance finite state machines then, identified a specification or properties to be satisfied by each module.

Now, we know what is the requirement of each module what attacks it needs to satisfy is perform? Or what property it needs to satisfy, we are going to identify those particular specification on properties for each and every modules. Then what we are going to do? We are going to use our verification techniques to check for the correctness of those particular properties. Once you get the confidence that yes whatever I have designed and whatever requirements I have specified that my design is going to meet all those particular requirements and properties. Then what will happen next we move for the implementation.

So, generally what will happen for hardware component we may use some hardware description language like your Verilog or VHDL? And we are going to get the implementation of those particular hardware component then we can simulate it we can look for the testing it then finally, we are going to implement it in our silicon. And the software programming for software component we are going to remain use some programming language to implement those things. On the other hand also sometimes what will happens? For some small system people come up with the initial design and they wrote the modules for hardware component they wrote down the modules in your hardware description language like Verilog and your VHDL; software component they use some programming language to implement may be c java python etcetera.

Then if it is a small system then what people used to do from those particular implementation they try to extract of the models ok. Already I think, I have mentioned that we are having mechanism to find out the behavior or find out the model. And on those particular model people used to apply the verification to check for the properties, this is basically for small system we can go this way also. But for because system generally we first verify our design then we go for the implementation.

(Refer Slide Time: 62:18)



Tools

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

Now, we have seen how to look for the correctness of a system that we are going to design ok. So, for that some of the tools I am just going to mention it if you are interested you can look for it; one is your CUDD. This is a binary decision diagram package developed at your university of Colorado at Boulder. So, CUDD is a BDD manipulation package. So, you this is for academic purpose it is available ok, it is a server. So, you if you are interested you can download it and you can see how we can use these things for your construction of BDD or manipulation of BDD for some Boolean function. Another one your nuSMV, SMV is this is your Symbolic Model Verifiers. This is nuSMV is the extension of SMV, where we are having the more features. So, this is the first model circuit based on BDD ok. So, this is also available.

So, if you are interested we can look for or you can download it install it and see how system will be specified in this particular nuSMV. How the properties will be specified then how they are going to check the while you are going to check the correctness then what are the outputs that we are going to get, you can look for it. Another one which is the bounded model checker this is some sort of bound is specified in a model checker; that means, well I am going to look for a particular execution trace how long I am going to explore the system ok. This is because we have to say this thing so this is going to specify some bound.

So, it is also a model checker only another one which is having spin this is a LTL model checker. So, basically we are talking discussing about CTL Computational Tree Logic, Computation Tree Logic and we have seen that I have got him how to check those particular formula. But LTL is another temporal logic which is your Linear Time temporal Logic. So, we are having a model checker called spin which take care of LTL. Now what is the difference between CTL and LTL? In case of CTL you just see it is a step formula in case of LTL it is a part formula. This is the basic difference. So, the truth values of CTL will be defined on a state of the system, but true values of LTL will defined over a part in the system.

So, this is the difference. So, with the LTL also you can specify our property and we are having a model checker for LTL properties which is known as spin. So, if you are interested you can just look for those particular tools and see how we can use those particular tools during your designing phase ok. With this I conclude my lecture here today.

Thank you all.