

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 - 20
Equivalence between CTL formulas

Hello everybody welcome back to the online course on Embedded Systems Design Verification and Test. So, in our last class we have introduce about the notion of temporal logic and we have seen that in temporal logic apart from the logical operator we are having some temporal operator. We have discussed about those temporal operators and the meaning of those temporal operators ok.

(Refer Slide Time: 00:56)

- Temporal Logic
 - Temporal Operator
 - Meaning of Temporal Operator

Now, today we are going to discuss on particular class of temporal logic, which is known as CTL: Computational Tree Logics. There are several verities or several classes of temporal logic and CTL is one of them. So, we are going to discuss about the syntax and semantics of CTL.

(Refer Slide Time: 01:20)

Temporal Logic

- Temporal Logic
 - Meaning is defined over a model.
- Given a model M and a temporal formula φ , we define an inductive definition for the notion of φ holding at a position S_j in M and denoted by $(M, S_j) \models \varphi$

Now, in temporal logic what we have seen that we are having temporal operator by which we can reason about the time, like whether always something is going to happen whether in future something good will happen like that and the truth values of temporal operator or temporal formula is given over a model. We have seen that given a model M and a temporal formula φ , we define an inductive definition of the notation of φ holding at a position S_i in M and it is denoted by $(M, S_i) \models \varphi$. It models φ or φ holds in state S_i of the model M , this is the notation of giving the meaning of a temporal formula.

Secondly already I have mention that temporal formula is valid in a state of a model or we can think that or we can say that temporal formula is valid or true in a part of a given model. So, we are having the notion of state formula and path formula ok.

(Refer Slide Time: 02:42)

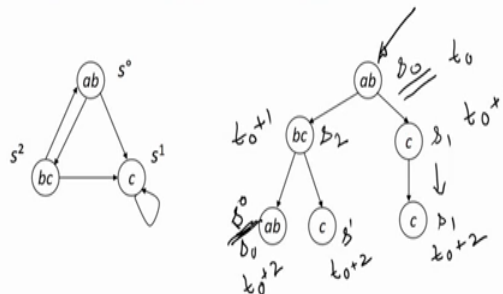
- Type of Formulas
 - Path Formulas
 - State formulas

In case of path formula the truth values of the temporal formula is defined over a path, in case of state formula the truth values of a temporal formula is define on a state of a given model. So, we must have a model and truth values of the temporal formula is defined either on state or over a path. In case of path formula truth values is defined over a path or in case of state formula truth values is defined in a particular state.

(Refer Slide Time: 03:21)

Temporal Logic

- Computational Tree Logic (Branching Time Logic)
 - Meaning is defined in a state over a model.



So, after getting the idea about the temporal operators and temporal logic, we have going to discuss about a particular class of temporal logic which is known as computational

tree logic CTL. On the other hand CTL is branching time logic, already we have mention about the notation of time progress. Either we can consider time progress in a particular execution trace which is basically known as your linear time logic. And, on the other hand there are several possible execution trace of a from a given state in a model and if we are going to reason about all possible execution traces then we have to deal with the branching time logic.

So, CTL is a branching time logic which is known as computational tree logic and a meaning of the CTL is defined over a model, already I have said that temporal logic has to be defined over a model. But, for definition of CTL we are going to consider a particular class of model and we will discuss about those particular model and a we will these discuss how we are going to define the truth values of where formula in that model.

Secondly CTL is a state formula the truth values CTL is defined in a state of a given particular model. So, just we have consider this particular model that it is a being free state s_0 s_2 and s_1 and it says that if it is in s_0 , then we can have 2 possible transition. Either it can go to s_2 or depending on the situation it can go to s_1 , one it is in s_2 then form s_2 either we can have a transition to s_3 or we are having a transition back to s_0 and once we come to this particular state s_1 then system remains in this particular s_1 .

So, this is this model is nothing but a state transition diagram, the system is having several state and it shows the transition between different states. So, here in this particular model we are having 3 different state s_0 s_1 and s_2 and if you look in to it then you will find a there are 5 different transition in this model. Now, this model can be expanded by looking into those particular transitions. Now those (Refer Time: 06:04) if we are in s_0 from s_0 we can go to s_1 or the you can have the transition to s_2 . Now when we are in transition s_1 then we can have a transition 2 s_1 itself ok.

Now, on the other hand when we are coming to s_2 we can go to the state s_0 or we can go to the state S_1 . Now in this particular case you just see that these state s_0 and the initial state s_0 are having a same behavior, the label a b indicates that we may have some proposition over here 2 proposition variable a and b the truth values of a and b is true at that particular state s_0 .

Now, when we look into the execution then that will be expanded to a particular tree. So, this step again we are encountering s_0 because we are having a go back path, so in this

particular state also property same that it is having the propositional formula a and b are true. But, only difference now we can consider over here the if this is starting point and we said this is your time t_0 , then after having one particular transition we can say that we are going to state is time $t_0 + 1$ or if the other possibilities is also there. So, time at that particular instance is your $t_0 + 1$, we just considering a onetime unit for one transition.

So, in this particular way now (Refer Time: 07:52) now when system progresses then in second transition I am going to get that the time of this particular state is $t_0 + 2$ here also $t_0 + 2$ and this particular state is also a $t_0 + 2$. Now, when we consider about this state s_0 and the other state s_0 here behavior is same because, some of the propositional variables are true in these particular states, but only difference is the time or you can say time step, it is showing the behavior of the system at time t equal to 0. But this particular state showing the behavior of the system or the state of the system at time $t_0 + 2$ after 2 time (Refer Time: 08:45), I am considering the time step for one particular transition is one.

So, we are just expanding the given system to a tree, this tree is known as computational tree and the logic define of a this tree to define some properties that is why the name is given as your computational tree logic. So, we are going to discuss about a syntax and semantics of this particular logic CTL or computational tree logic.

(Refer Slide Time: 09:20)

Syntax of CTL

A CTL formula comprises

1. Atomic propositions such as $\{p, q, r, \dots\}$
2. Path Quantifiers $\{A, E\}$
 - a. A : all paths starting from a given state.
 - b. E : there exists at least one path from a given state.
3. Propositional logic operators such as AND (\wedge), OR (\vee), NOT (\neg)
4. Temporal operators $\{X, F, G, U\}$
 - a. NEXT: next states of current state.
 - b. FUTURE: any one of future states from the current state.
 - c. GLOBAL: all future states from the current state.
 - d. UNTIL: Some CTL formula holds until another CTL formula, from the current state.

*P: TRUE
FALSE*

*X, G, F
U*

Now, what is syntax? Now, as you know that in every logic or in every computer programming language we have to define the syntax and according to the syntax we have to write valid sentences of that particular logic or of that language. So, when we look into the CTL formulas, then CTL formula comprises of one component atomic proposition. So, we are having some atomic proposition $p \ q \ r$ then if you consider one particular atomic proposition, then what will happen the truth values of this particular atomic proposition will be either true or false.

So, we are having some atomic proposition the truth values of the atomic propositions are either true or false depending its states. Along, with that we are having one particular notion or particular identity which is known as your path quantifier. So, it is quantified over a paths, so this path quantifiers is A and E, A stands for in all possible path and E stands for their existed path.

So, we are going to reason about in all possible paths or there may exist a path, if we follows that particular path a property will be true on that particular state, so these are a path quantifier A and E. As we have already mention that all the propositional connective can be used in our temporal logic, so access all the propositional operator can be used in our CTL also. So, these are all basic operators I am saying AND OR and NOT similarly others also a expressive or NAND NOR everything can be used in CTL and we are having temporal operators and the basic temporal operators are basically NEXT which is represented by X, FUTURE which is represented by F, GLOBAL which is represented by G and having an UNTIL operator which is basically until.

Out of this particular 4 operators already we have discuss you will find that NEXT, GLOBAL and FUTURE, these 3 are unary operators; it works only on one particular variable. But UNTIL U is a binary operator we need 2 operators to express the formula or sentence using this particular until operator. So, already we have discussed about the many of this particular co operators. So, in CTL we are going to use this particular 4 CTL operator sorry temporal operator. So, our CTL formula consist if this particular 4 temporal operators also. Now, the formal syntax of CTL can be given in the BNF notation at in all of you know about the BNF notation. So, this is the creeps going to representing the syntax of a given language.

(Refer Slide Time: 12:39)

We can define CTL formulas as:

$$\Phi ::= \perp \mid \top \mid P \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid \underline{AX}\phi$$

$$\mid \underline{EX}\phi \mid \underline{AF}\phi \mid \underline{EF}\phi \mid \underline{AG}\phi \mid \underline{EG}\phi \mid \underline{A}[\phi \ U \ \phi] \mid \underline{E}[\phi \ U \ \phi];$$

where

- The symbol \top means truth value 'true' and symbol \perp means truth value 'false'.
- P ranges over a set of atomic propositions

Handwritten notes:
 ϕ
 $\phi \oplus \phi$
 \perp : bottom FALSE
 \top : top TRUE

So, in CTL: Computation Tree Logic the BNF notation of the CTL is as follows, we are using 2 symbols one is known as your bottom and second one is your top ok. So, this is basically radius bottom and second one is top. Basically bottom we write bottom to indicate the truth value false and top we will write the symbol top to represent a truth value true. So, truth value true and truth value false are also CTL formula, so top and bottoms are CTL formula.

Secondly we are having a set of atomic proposition. So, every atomic proposition is a CTL formula, so whatever atomic proposition we are using in our system so every atomic proposition is treated as an CTL formula. Independently it is a formula it is a CTL formula because, truth values of this formula will be either true or false. If it is true in a particular state then we said that a truth values of that particular atomic proposition or the CTL formula is true at that particular state. Similarly, if the truth value is false of that atomic proposition then we said that the corresponding CTL formula is false at that particular state.

Now, if phi is a CTL formula, so then not a phi is also a CTL formula basically we are using this particular logical connective phi and phi is also CTL formula phi or phi is also a CTL formula phi implies phi is also a CTL formula or you can list any other logical connective. So, if I am having a logical so if I am having a CTL formula phi then phi with any logical connective, phi is also a CTL formula this may be conjunction,

disjunction or explosive or implication whatever it may be it will become a CTL formula.

Now, we are having 4 temporal operators, so with the help of this 4 temporal operator also we can construct CTL formula. So, these are the CTL formula so X next is an CTL operator temporal operator. So, with act we are constructing to CTL formula AX phi and EX phi. So, AX phi basically says that in all possible path in the next state phi holds or EX phi basically we can say there exist a path at least there should be one possible execution path in which in the next state phi is true. So, we are getting 2 CTL formula from the temporal operator X AX and EX.

Similarly, we are going to get 2 CTL formula with a temporal operator F AF phi and EF phi in all possible path or there exist a path. Similarly in case of G operator global operator we are also going to get 2 CTL formula AG phi and EG phi like that for until operator. Also we are going to have 2 formula A phi until psi or I can say that the phi until phi and E phi until phi, one says that in all possible execution trace phi remains true until phi becomes true ok. Similarly, E says there exist a path, so these are the path quantifier and the path quantifier is used along with the temporal operators to form a CTL formula.

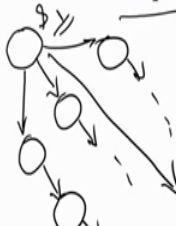
(Refer Slide Time: 16:46)

Let V be a set of atomic propositions

CTL formulas are defined recursively:

Every atomic proposition is a CTL formula

If f_1 and f_2 are CTL formulas, then so are $\neg f_1, f_1 \wedge f_2,$
 $AX f_1, EX f_1, A[f_1 U f_2]$ and $E[f_1 U f_2], AG f_1, EG f_1,$
 Aff_1, EFF_1



$A(f_1 U f_2)$
 $E(f_1 U f_2)$

So, basically we all said a let V be a set of atomic proposition done this is a component and every atomic preposition will treated as a CTL formula and this CTL formulas can

be defined recursively and every atomic proposition is a CTL formula. Firstly if it is a CTL formula then weight logical connective we can form CTL formula and with temporal operator we can form CTL formula. But if you look into it if you notice that every CTL operator is preceded by a path quantifier.

So, that is why if I am saying that f_1 until f_2 until is a temporal operator it says a f_1 remains true until f_2 remains true. So, this is a temporal operator and the truth values of this temporal operator basically define over a path it is a path formula. But if we give the path quantifier either A or E then it becomes a state formula, the truth values of this formula is defined for a state. So, basically I can say that if this is the state s_1 I can have several possible execution trace.

Now, if we are going to have this things now if I say that AF until f_1 until f_2 if this is CTL formula, it says that in all possible execution trace found is particular state 1 f_1 remains true until we are reaching state where f_2 is true. So, this behavior should reflect in all possible path then only we can say that $A f_1$ until f_2 is true.

Similarly, if we say $E f_1$ until f_2 then what it says at least we must have one possible execution trace where f_1 remains true until f_2 becomes true ok. So, with this particular path quantifier A and E we are constructing a state formula and the truth values of this particular temporal CTL formulas define in a state, so it is a CTL is a state formula.


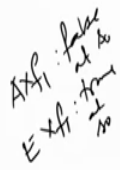
(Refer Slide Time: 19:30)


$AX f_1$ means: holds in state s^o iff f_1 holds in all successor states of s^o

$EX f_1$ means: There exists a successor such that f_1 holds


$A[f_1 U f_2]$ means: always until, in all paths such that f_1 holds until is f_2 satisfied.

$E[f_1 U f_2]$ means: There exists a path such that f_1 holds until is f_2 satisfied.



$E[f_1 U f_2]$



$AX f_1$

Now, what is a meaning of those particular things? Already, I have mention $AX f_1$ that means X is the next operator. So, AX and $EX AX$ is the their in all possible execution path in the next state f_1 is true and $EX f_1$ mean their just at least one path, where in the next state f_1 is true. If I look in to this particular scenario, so this is the state s_0 and if I set a f_1 , f_1 is label. What does it means it says that, in this particular state s_1 formula f_1 is true and in the state s_3 formula f_1 , but in s_2 formula f_1 is not true. So, this is the label basically we give we indicate the truth values of the formula, if it is true generally we indicate this particular formula to show that the truth value is true.

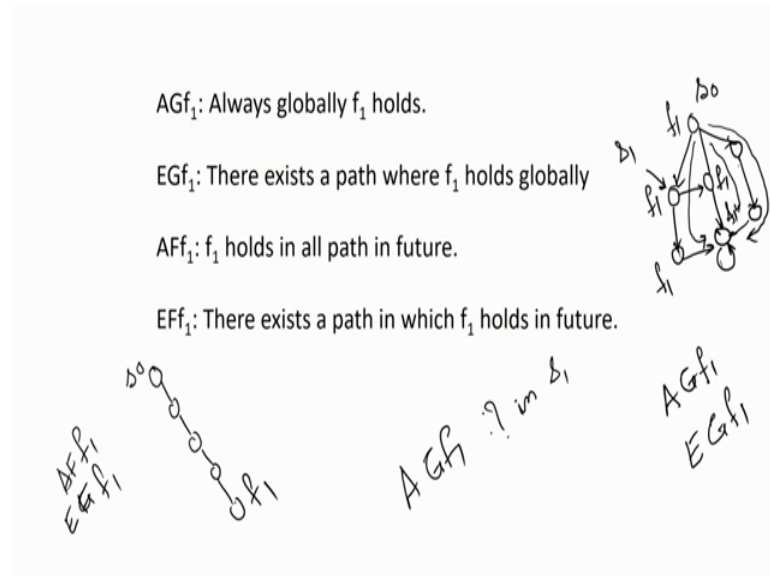
So, in this particular case you just see them at least we are having one particular part in the next step f_1 is true. So, in s_0 $EX f_1$ is true or you say that in the state s_0 $AX f_1$ holds because, we are getting one next state that means one next state when f_1 is true. But if I look into this particular scenario then $AX f_1$ in all part in the next step f_1 is true we are just looking for this particular possible behavior.

So, in that particular case $AX f_1$ is not true so $AX f_1$ is false at s_0 . But $EX f_1$ is true at s_0 because, in we are having 3 possible execution path from this particular state s_0 , but if we follow this particular path then what will happen in next state f_1 is not true. So, like that we have to see whether the truth values update is given temporal formula is true over they are all not in a particular state.

So, similarly $A f_1$ until f_2 and EF_1 until f_2 , so f_1 remains true until f_2 becomes true. So, in a particular path I can say that if the scenario is something like that, so that in this particular state is you know if I go by this particular path I will find that f_1 remains true until f_2 becomes true in a particular state. So, in this particular path this formula f_1 until f_2 is true, so if I come to this particular state s_0 .

So, here I can say that $E f_1$ until f_2 is true, like that from f_0 you may have several possible execution trace. So, if in all execution trace the behavior is something like that similar to this one, then I can say that in all path f_1 remains true until f_2 becomes true ok. But if anyone of this particular path these behavior is not showing or not satisfying then $A f_1$ until f_2 is false at that particular state s_0 .

(Refer Slide Time: 23:12)



So, similarly we are having the global, so in global it says that globally it is true; that means, in all possible state f_1 must be true then only we will say that $AG f_1$ is true if it is all possible path. So, if I am going to have some scenario or something like that, so if I look into this particular scenario then we will find that if I go if I proceed through this particular execution path and we will find that in all states f_1 is true.

Similarly, if I proceed by this particular exhibition trace then again we will find that in possible in all states f_1 is true, but if I follow this particular execution trace in this 2 state f_1 is not true. That means, if I am going to look for the truth values of this formula $AG f_1$ then $AG f_1$ is not true at s_0 because, we are going to get one execution trace where f_1 is not true in 2 of the states.

But if I say that $EG f_1$ here there exist a path globally f_1 holds yes, we can say that either we can consider this particular path or we can consider this particular path then in all the state f_1 is true. So, at least there exist a path where f globally f_1 holds, so we can say that $EG f_1$ is true. Now, in the same model now you consider this particular state s_1 , whether $AG f_1$ holds are true in s_1 . So, in that particular case you just see that from s_1 I am starting and it is having only one execution path in all the states f_1 is true. So, we can say that so here $AG f_1$ is true. Now, if I extend this particular give one more transition from s_1 to this particular state. Here also we will find that it is having 2 different execution possible trace, so in both the paths in all the state f_1 is true.

So, we can say that $AG f1$ is true in state s_1 , but already we have seen that $AG f1$ is not true in s_0 . So, already I have mentioned that truth value is defined in a state in case of state formula. So, similarly we are having the operator future F so again we are having AF and EF , so in all path in future or in all there exist a path in future.

So, it is like that if I am going to consider one particular state we may have a execution trace and we are going to say (Refer Time: 26:56) somewhere in future $f1$ hold some not. If this is the (Refer Time: 27:02) and we can say that in the state s_0 $AF f1$ is true. So, these are the operators and these are meanings and in case of CTL we are defining the truth values of a CTL formula in a state it is a state formula.

(Refer Slide Time: 27:20)

- In CTL, every temporal operator must be preceded by a path quantifier.
 - State formula

Now, what is the basic notion? If you see or if you observed you will find that every CTL operator is preceded by a path quantifier; that means, if we are having any CTL operator if it is preceded by a path quantifier, then we are going to get a state formula and CTL is a state formula all CTL are state formula.

So, that is why your if you look into the definition you just see these are the operator temporal operator that we are having, so all temporal operators are preceded by a path quantified either A or E and with the help of this path quantifier we are getting the state formula. So, truth values of CTL formulas is defined over a state and from observation what we can see all CTL operators are preceded by a path quantifier either A or E . So, we

are getting a state formula and the truth values of CTL formulas are defined in a state of a given model.

(Refer Slide Time: 28:35)

Examples

- $AG(p \rightarrow \neg EG \neg q)$
- ✓ $EGp \wedge E(q U r)$
- ✓ $AG \neg(p \wedge q)$
- ✓ $AG \neg(EF p \wedge q)$
- ✓ $AF EG p$
- ✓ $A[p U A[q U r]]$
- ✓ $A[AX \neg p U EX(\neg p q)] \rightarrow A[p U \neg q]$

Now, some example you just see if I am writing some example over here, as a notation you see that we are using those particular path quantifier A or E or the temporal operators and the logical connective. Now, whether these are valid CTL formulas or not we have to check for it and if it is syntactically correct, then we say these are the well form formula of that particular logic.

Now, consider one particular example over here, so I am talking about $AG p$ implies not of EG not of q . So, whether it is a valid CTL formula or not constructively ok. Now in this popular case now you see the BNF notation of construction of CTL formula and see whether we are going to get CTL formulas or not. So, in that particular case you just see that p is a atomic proposition, so every atomic proportion is a CTL formula. So, this component is a CTL formula, q is also a CTL formula because it is also a atomic proposition. If ϕ is a CTL formula the not of ϕ is also a CTL formula, so what I can say not of q is a CTL formula.

Now, we can use if ϕ is a CTL formula then $EG \phi$ is also a CTL formula. So, $EG \phi$ is also a CTL formula because globally there exist a path globally not of q holds I can be note that if ϕ is a CTL formula not of ϕ is also a CTL formula, so negation of this is also a CTL formula. Now, if I am having 2 CTL formula ϕ_1 and ϕ_2 then we can

connect this 2 CTL formula with any logical connective. So, this is also a CTL formula p implies not of EG not of q . Now if ϕ is a CTL formula then with the help of temporal operators along with the path quantifier we are going to get a CTL formula, so $AG \phi$ is also a CTL formula. So, if we look into the component wise then we can say that these are all the components has CTL formula and by connecting them with logical connective or temporal operators we are going to get a valid CTL formula.


So, in this particular case now we are we have seen I can say that p and q these are sub formulas of this given formula. Again since q is a formulas or not of q is also a sub formula of this given formula. Similarly EG not of q is also a sub formula of the given formula, similarly not of EG not of q is also a sub formula of this given formula and finally we can say that p implies not of EG not of q is also a sub formula. So, in this particular way so finally this sub formula is connected with AG in all path globally, so the given formula is also a CTL formula.

So, like that if you analyze the other formulas that I have mention over here, you will find that all of these are constructively correct CTL formula they are correct CTL formula. So, here another example you just see that here p and q are atomic proposition they are CTL formula so p and q is also CTL formula. So, if ϕ is a CTL formula then $EF \phi$ is also a CTL formula, if ϕ is a CTL formula then negation of ϕ is also a CTL formula, if ϕ is a CTL formula then $AG \phi$ is also a CTL formula. So, like that you can look for all those particular equation that whatever I mention and you will find that all are constructively correct CTL formulas.

(Refer Slide Time: 33:15)

Examples

- Gp
- $EFGr$
- $F[r U q]$
- $AEFr$
- $A[(r U q) \wedge (p U r)]$



Now, look another set of formulas, so in this particular case just see that if I am going to write this particular formula $F r$ until q . So, what it means in future r remains true until q becomes true whether it is a CTL formula or not. It is a do not confuse it is a temporal formula no doubt about it because, we are using temporal operators over here what are the temporal operators that we are using f and until future and until, so $F r$ until q .

Now, in this particular case say r and q are atomic proposition so we can treat them as your CTL formula, when I come to r until q then it is a temporal formula no doubt but it is not a CTL formula because, this until operator is not preceded by any path quantifier and after I am getting $F r$ until q again it is a temporal formula. But it is not a CTL formula because again f is not preceded by any temporal operator so it (Refer Time: 34:33) this things. So, in this particular case what I can say if I write $E F E r$ until q , so that means there exist a path r remains true until q becomes true. So, now this until operator is preceded by this particular path quantifier is, so this is a CTL formula.

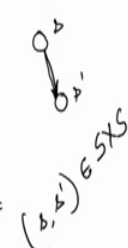
But I am having another formal temporal operator have which is not preceded by any path quantifier, so it is also not a CTL formula. But if I write $A F E r$ until q then you will find that now these becomes a valid or correct CTL formula because this temporal operator until it is preceded by path quantifier E and this particular f temporal operator is again also preceded by another path quantifier A . So, it becomes a CTL formula, but access the given formula are $F r$ until q is not a CTL formula. So, if I or if you look or

analyze these particular formulas you will find that these are not correct CTL formula, but all of these are temporal formulas but not CTL formulas.

(Refer Slide Time: 35:55)

Temporal structures

- The semantics of CTL is defined over a model M , which is defined as 3-tuple $M = (S, \rightarrow, L)$
- **Definition:** A temporal structure $M := (S, \rightarrow, L)$ consists of
 1. A finite set of states S
 2. A transition relation $\rightarrow \subseteq S \times S$ with $\forall s \in S \exists s' \in S: (s, s') \in \rightarrow$
 3. A labeling function $L: S \rightarrow \wp(V)$, with V being the set of propositional variables (atomic formulas)
- This structure is often called Kripke structure.



Now, we are going to look for the definition of semantic of temporal formulas, how to define the semantics of a temporal formula? So, already we have said that their truth values of a temporal formula defined on a model, so in case of CTL we are going to consider a particular model. So, what is this model the minimum component that we have in a model is having a 3 tuples S arrow and L the basic notations that we are using over here.

So, the semantics of CTL is defined over a model which is define as 3 tuple, so M is basically consist of S arrow and L now what is this particular component S is nothing but the finite set of state. So, you are going to define the truth values of a CTL formula on a finite state systems or number of states are finite.

We are having a transition relation arrow which is a subset of Cartesian product S cross S ; that means, if we having set of states. That means, we are having a transition from any state to any state with for all S belongs to the set of state S , there exists s dash belongs to S such that s and s dash belongs to this particular transition relation. That means, if this is s and I am having another state s dash and if I am having a transition from s to s dash. So, the member s to s dash this is nothing but a member of S cross S Cartesian product of s on s .

So, this member is a member of this particular transition relation if we are having an transition from state s to S and the basic emphasis it is given with this particular state particular symbol for all state. So, basically for all state there is must be next states that means from every state we are having an outgoing transition. If we do not have the outgoing transition then this is not a valid CTL structure or temporal structure for defining CTL formula and along with that we are having a labeling function which is known given as $L S$.

So, what is this it is L is a function from the set of states to the power set of V and what is V now V is nothing but the set of atomic proposition because, we are going to work with a set of atomic proposition the truth values will be either true and false, so we are having a labeling function. If a particular atomic proposition is true in a particular state, then we label this particular state with the help of that particular atomic proposition.

So, you just see the CTL structure is similar to finite state transition machine, we are having a finite number of state and we having some transition from states to states of this particular machine. But the transition is having a particular property it is complete, that means form every state there must be an outgoing yes or outgoing transition to some next state, along with that we are having this particular labeling function.

So, this is extra apart from our final state machine then formal definition of final state machine we are having this particular labeling function. So, once you have distinct then generally in historical region this is known as our Kripke structure; that means, the truth values of a CTL formula is defined in a Kripke structure.

(Refer Slide Time: 40:12)

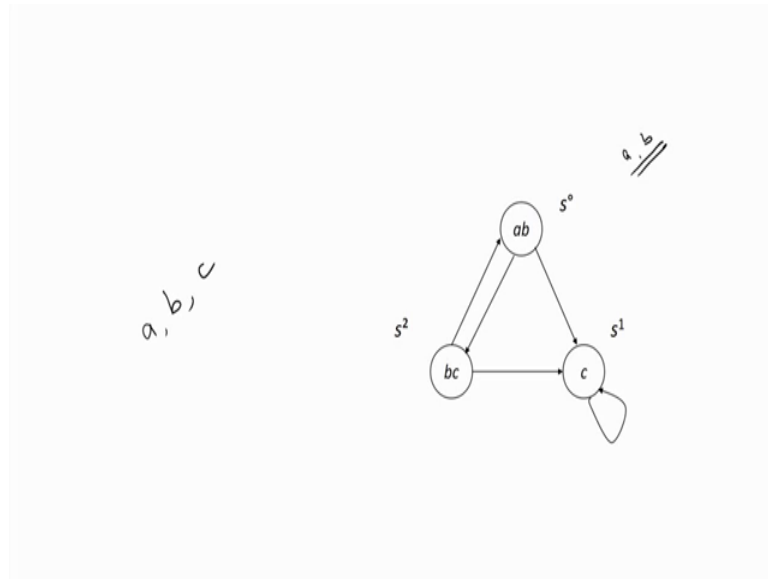
Semantics of CTL

- This model is also known as Kripke structure.
- A Kripke structure is similar to a state transition diagram, with
 - All states must have at least one outgoing edge.
 - Each state is labeled with one of the element of the power set of atomic propositions.

So, what is a Kripke structure basically now I am saying that it is similar to the finite state machine, but transition relation is having a special behavior it is complete in nature. That means, every state should have an outgoing that is means every from every state we should have a transition to some (Refer Time: 40:35) and along with that we are having a labeling function if we are working in a particular system.

Then if the atomic propositions are true in a particular state then that is state will be label with the help of this particular labeling function. So, these are the 2 extra things that we are having along with a finite state machine. So, this model is known as our Kripke structure and the meaning of a CTL formula is defined in a Kripke structure.

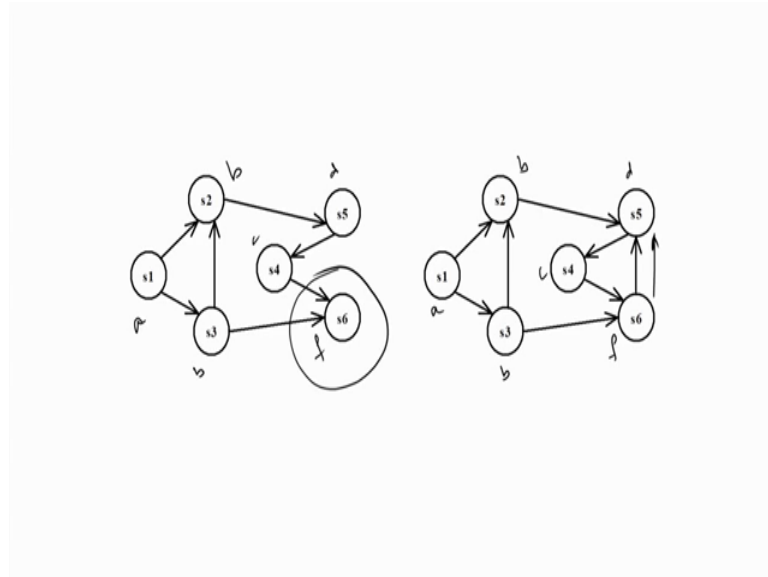
(Refer Slide Time: 41:10)



Now this is simple example you just see if you look into it, now what does it means it is having 3 state s^0 s^1 and s^2 and basically it is having 3 atomic proposition, we are having 3 atomic proposition a b c and when we are designing the system and when we are abstract of the model, it seems that in the state s^0 the atomic propulSION a and b are true. So, that is why this is the labeling function we have label is particular state with a and b , so a and b are true in this particular state s^0 .

Similarly, the atomic proposition b and c are true over state s^2 , so it is labeled with b and c . Similarly in state s^1 only atomic proposition c is true so this is labeled with c . So, this is the labeling function with the help of labeling function we have labeled those particular states and secondly we have the transition say s^0 . We are having a 2 outgoing as a 2 transition from s^2 we are having 2 transition, from s^1 we have one transition to itself. That means, every states is having an ongoing transition, so this is a valid Kripke structure and we can define the meaning of CTL formula in this particular model.

(Refer Slide Time: 42:37)

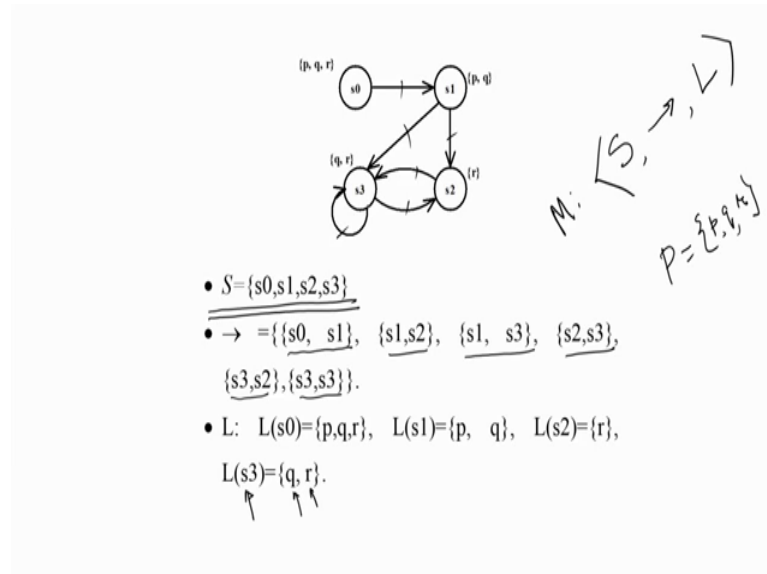


Now, just consider this 2 randomly I have drawn 2 figures whether these are Kripke structure or not or here I am not talking about a labeling function if I label them with some atomic proposition, then it is fine I can say that d f. So, a b b c d f so labeling function is there that it is label with the atomic proposition that we are going to work with.

Now, whether these are valid Kripke structure or not, so one is there labeling function is there now we have to see the transition function whether it is complete or not. So, in the first diagram if you just notice when we come to this particular state s 6, we will find that there is no outgoing s ok, there is no outgoing transition from this particular state s 6. That means, transition relation is not complete so it cannot be considered as a Kripke structure.

But here is also I am putting one extra s over here, so what it says I am from s 0 I am having transition to s 5. Now if you observe you will find that from every state we are having at least one outgoing transition. So, this is a valid Kripke structure and we can define the meaning of CTL formula on this particular model.

(Refer Slide Time: 44:01)



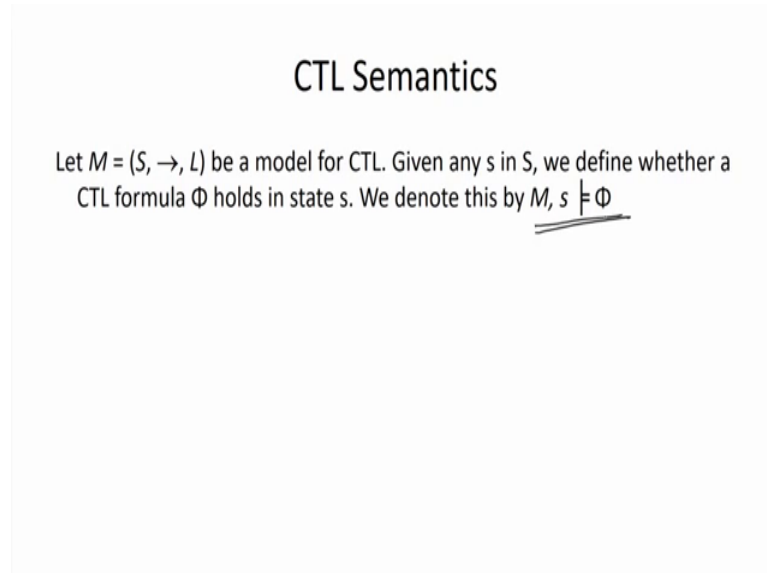
Now, as a simple example you just see this is a finite state machine we are having 4 state s_0 s_1 s_2 and s_3 . So, this is the set of state first component as because it defines it a model as your s transition relation and labeling function, this is the model M for Kripke structure. So, state s is nothing but s_0 s_1 s_2 and s_3 , what is the transition relation you just see as there is a transition from s_0 to s_1 .

So, a s_0 to s_1 is a component from s_1 we having to transition one is going to s_3 and one is going to s_2 , so s_2 s_1 to s_2 and s_1 to s_3 . When we are in s_2 then we are having a transition from s_2 to s_3 and when we arrive that s_3 then we having a transition back to s_2 . So, s_3 to s_3 and there is a self loop self transition so s_3 to s_3 . So, in this particular transition relation we are having these particular sixth components that means it is having 6 different transitions, so 1 2 3 4 5 6 so this transition relation is having these 3 transition.

Now, what is the labeling function by looking into the labels, you can say that the labeling function is defined like that labeling function at the state s_0 is nothing but p q r . So, if I am using a set of atomic proposition as say p q r ok. So, in that particular case you find that the labeling function next state s_0 is your p q r , similar label of s_1 is your p and q labeling function at s_2 is your r and labeling function at S_3 is your q and r . That means, it says that state s_3 is label with the atomic proposition q and r and its indicated

that the atomic proposition q and r are true in this particular state S 3 ok, this is the notion about the truth values of atomic proposition.

(Refer Slide Time: 46:22)



CTL Semantics

Let $M = (S, \rightarrow, L)$ be a model for CTL. Given any s in S , we define whether a CTL formula Φ holds in state s . We denote this by $M, s \models \Phi$

Now, how we are going to define a semantic so let M be a model, so M is having a component set of state the transition relation and the labeling function be a model of a CTL formula given any s in S . We define whether a CTL formula ϕ holds in state s , we denote this by $M, s \models \phi$, on the other way we can say that ϕ the CTL formula ϕ holds in the state s of the given model M . Now, we are going to formally define the semantics of each and every CTL formulas.

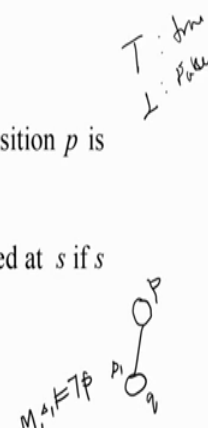
(Refer Slide Time: 47:00)

The relation $M, s \models \varphi$ is defined by structural induction on φ , as follows

$M, s \models \top$ and $M, s \not\models \perp$;

$M, s \models p$ iff $p \in L(s)$; atomic proposition p is satisfied if label of s has p .

$M, s \models \neg\varphi$ iff $M, s \not\models \varphi$. $\neg\varphi$ is satisfied at s if s does not satisfy φ .



So, we are going to define it on a model M and the structural induction on φ because, φ is a CTL formula which consists of several sub formulas. If all the sub formulas truth values of all the sub formulas is defined in a state, then only we can define the truth values of the given CTL formula in that particular state.

So, what are the notion it says that in state s in model M it models the top, top is nothing but true already I have mention and bottom is nothing but false. That means, it says that true is always true in all the state and false is always false in the in all state this is the basic notion about the truth values. So, basically it says that in all state the true is true or top is true truth values of true is true and truth values of false is not true. So, it say says that it does not model φ ok, this is about the truth values true and false which are basically constant in our logic family.

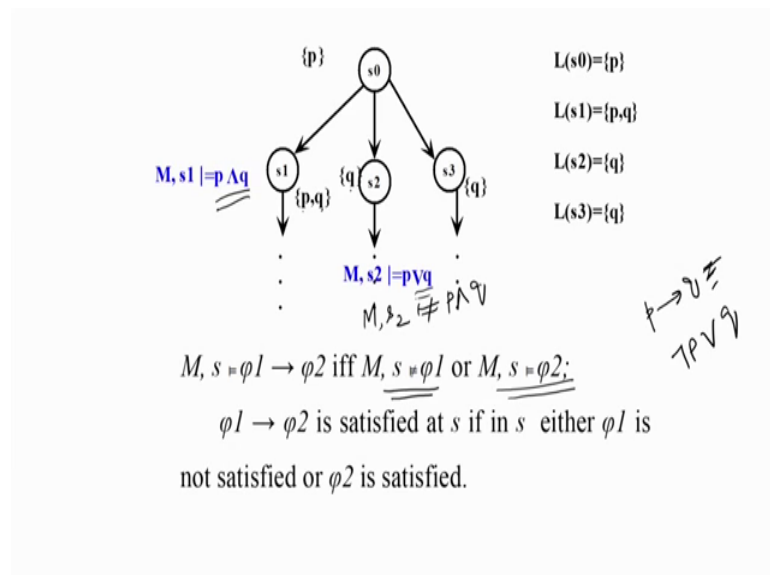
Now, again it says that whether in any state of the given model the atomic proposition is true or false, it says that if p belongs to the labeling function of that particular state s then an s models p or we can say that p is true in the state s of the model M . On the other hand if it is not a member of this particular labeling function, then truth values of this particular atomic proposition is false at that particular state ok.

So, atomic proposition is considered to be true in a state provided it is a member of the labeling function of that particular state. Similarly it says the $M, s \models \varphi$ not of φ

that means, not of phi is true in the state s provided phi is not true at that particular state s.

So, $M \models \phi$ does not model phi so in the particular case we say that $M \models \text{not of phi}$ this is very simple I think you understand. So, if this is a state and if it is labeling function is model $s \models p$ and here you say it is q , so it is p is not labeled at that particular state. So, here I can say that in this particular model M instead this particular say s_1 it models not of p , because p is not true at that particular transition. So, it says that it models not of phi provided it does not model this phi.

(Refer Slide Time: 50:05)



So, these are the small example I am giving, so these are the labeling function L and $s_0 \models p$ and $L(s_2)$ and $L(s_3)$. So, this is $M \models p$ but $M \not\models p$ because it is not labeled to it p .

(Refer Slide Time: 50:22)

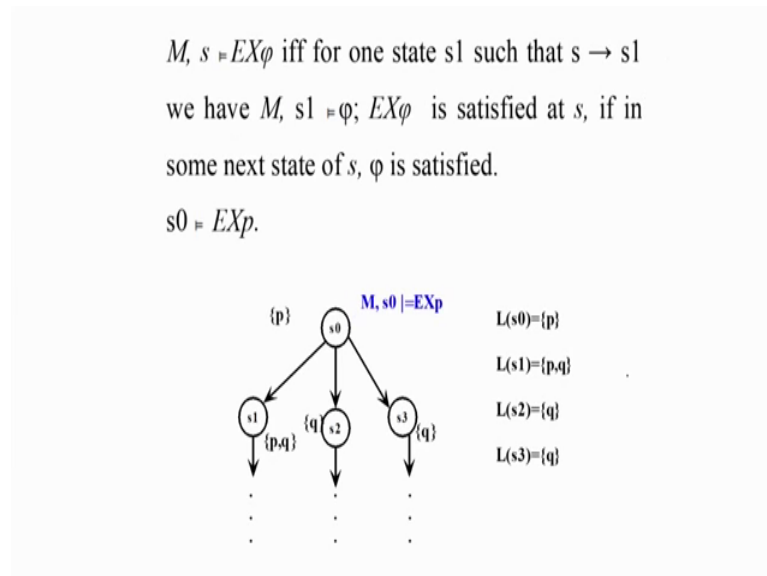
$M, s \models \varphi1 \wedge \varphi2$ iff $M, s \models \varphi1$ and $M, s \models \varphi2$;
 $\varphi1 \wedge \varphi2$ is satisfied at s if in s both $\varphi1$ and
 $\varphi2$ are satisfied.
 $M, s \models \varphi1 \vee \varphi2$ iff $M, s \models \varphi1$ or $M, s \models \varphi2$;
 $\varphi1 \vee \varphi2$ is satisfied at s if in s either $\varphi1$ or $\varphi2$
is satisfied.

So, similarly now we can use any logical connective it is very simple say if it is \wedge connective, it says that in a model in the state s of the model M $\varphi1$ and $\varphi2$ holds or not, whether $\varphi1$ and $\varphi2$ is true or not provided $M, s \models \varphi1$ and $M, s \models \varphi2$ ok. So, if both $\varphi1$ and $\varphi2$ are true in this particular state then we can say that this models $\varphi1$ and $\varphi2$.

Similarly, we know the or connectives either $\varphi1$ is true or $\varphi2$ is true, that is why it say that the other iff $M, s \models \varphi1$ or $M, s \models \varphi2$ and we can say that $M, s \models \varphi1$ or $\varphi2$. So, these are the some connectives with an $M, s \models \varphi1$ implies $\varphi2$. So, we know the equivalence p implies q is equivalent to not p or q , so it says that if M, s does not model q 1 and $M, s \models q$ 2. So, not of p or q so if not $\varphi1$ is not true in s and $\varphi2$ is true in s then we can say that $M, s \models \varphi1$ implies $\varphi2$.

So, this is small example I can same notation that I am using, so $M, s \models p$ 1 or p q p and q because both p and q is true over here. So, in s 1 p and q is true, but here only q is true, but still $M, s \models p$ and q is true. But M, s 2 does not model p and q am I right because in next only q is true but p is not true. So, p and q is false so M, s 2 does not model p and q .

(Refer Slide Time: 52:24)



Now, the logical connectives is simple now we have to look for the temporal connectives. So, one is your whether in M s model AXq , so it says that M s model Xq or I can say that $AX\phi$ is true in a state s of the model f . Provided in all s_1 such that there is a transition from s to s_1 we have $M s_1 \models \phi$, that means we are going to consider all the possible transition from this particular state s M . Whatever next that we are going to get in all the next state ϕ is true $M s_1 \models \phi$, then we can say that $M S \models \phi$ $AX\phi$.

So, in this particular case you just see whether $M s_0 \models AXq$, so in all path in the next state whether q holds or not. So, from S_0 I am having 3 next state, so s_1 q is true s_2 q is true and S_3 also q is true. So, yes it models if I check whether and $s_0 \models AX q$ yes it is true because in all the next state q is true.

So, similarly EXq whether $M s \models EXq$, so here also we are going to look for a transition from s to s_1 and if we are going to one such transition where $M s_1 \models \phi$ then we can say that $EX\phi$ is true in that particular given state. So, AX says that you look for all the next states and check whether q or ϕ is true in that particular states or not are for $EX\phi$ you look what is one such next state where ϕ is true and $s_1 \models \phi$. So, for one state s_1 such that we having a transition from s to s_1 we have $M s_1 \models \phi$ then we can say that $M s \models EX\phi$.

(Refer Slide Time: 54:42)

$M, s \models AG\phi$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3$
 $\rightarrow \dots$, where $s = s_1$, and all s_i along the path,
 $M, s_i \models \phi$. $AG\phi$ is satisfied at s if all states of all
 paths from s satisfies ϕ .

$M, s_0 \models AG q$

$L(s_0) = \{p, q\}$
 $L(s_1) = \{p, q\}$
 $L(s_2) = \{q\}$
 $L(s_3) = \{q\}$

M, s models $AG \phi$ it holds if all path, we are going to consider all paths such that s_1 to s_2 to s_3 like that we having s , but all paths where s equal to s_1 . So, path starting from s_1 and this s_1 happens to the s , then along this particular path all s_i along the path M, s_i models ϕ ok. So, first you consider all possible paths now in all possible path you consider all the states s_i and if in all s_i that M, s_i models ϕ then we say that A, s models $AG \phi$ ok. In all possible path in all state if ϕ is true then M, s model $AG \phi$ or we can say that $AG \phi$ is true in the state s of the (Refer Time: 55:48) model M ok.

So, there is a transition models it says that whether in whether it is true M, s_0 model $AG q$. So, you just see that we are talking about the s_0 , in s_0 q is true I am having one transition like that. So, in all the in this particular path in all the states q is true, I am having another transition like that. So, in this transition also q is true and all those particular q is true and this is the another transition we are having.

So, here also in all states q is true, that means from s_0 if we consider all such path and in all s_i along the path if it models ϕ then we say that $AG \phi$ is true, so in all state in possible path q is true. So, in s_0 $AG q$ is true that means M, s_0 models $AG q$ So, this is similarly $EG q$ so it is with the similar notion we can say that if there is a path.

(Refer Slide Time: 57:04)

$M, s \models AF\phi$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$
 ., where $s=s_1$, and for at least one s_i along the path,
 $M, s_i \models \phi$. $AF\phi$ is satisfied at s if some “future” state
 of all paths from s satisfies ϕ .

$M, s_0 \models AFq$

$L(s_0) = \{p\}$
 $L(s_1) = \{p, q\}$
 $L(s_2) = \{q\}$
 $L(s_3) = \{q\}$

Now in the $AG\phi$ I am talking about for all paths but here we are taking a there is a path and remaining condition is same that in all s_i along the path. Now you are considering a particular path and in all states of that particular path if ϕ holds, then we say that M model $EG\phi$ $AF\phi$ again in all path in future. So, again you say that if all paths we consider all such type of paths where s is equal to s_0 the starting state of the path and for at least one s_i .

So, we can consider this particular path and you consider all possible path and along this all possible path at least we are going to get one state s_i , such that $M s_i$ model ϕ then you can say that in all path $AF\phi$ is true. So, consider all possible path of this sort and along all path at least loop for at least one s_i ok, such that $M s_i$ models ϕ that means ϕ holds in the state s_i then we say that $M s_i$ model $AF\phi$ a meaning is like that $AF\phi$ is true in the state s of the model M .

(Refer Slide Time: 58:32)

$M, s \models EF\phi$ holds iff there is one path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where $s=s_1$, and for at least one s_i along the path, $M, s_i \models \phi$.

$M, s_0 \models EFp$

- $L(s_0) = \{q\}$
- $L(s_1) = \{p, q\}$
- $L(s_2) = \{q\}$
- $L(s_3) = \{q\}$

So, similarly we can look for EF phi, so in case of EF phi it says that it is similar to EF phi, but it says that if there is one path now we are not going to look for all path we are going to consider one path. If there exists a path such that at least we are going to get one s_i along that particular path, such that $M, s_i \models \phi$ then you can say that $M, s \models EF\phi$. So, this is the exact semantics of EF phi.

(Refer Slide Time: 59:05)

$M, s \models A[\phi_1 U \phi_2]$ holds iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$, where $s=s_1$, $\phi_1 U \phi_2$ is satisfied, i.e., there is some s_i along the path, such that $M, s_i \models \phi_2$, and for each $j < i$, we have $M, s_j \models \phi_1$.

$M, s_0 \models A[p U q]$

Handwritten notes: $s_j \models \phi_1$, ϕ_1 , ϕ_2 , ϕ_1 , ϕ_2 , ϕ_1 , ϕ_2

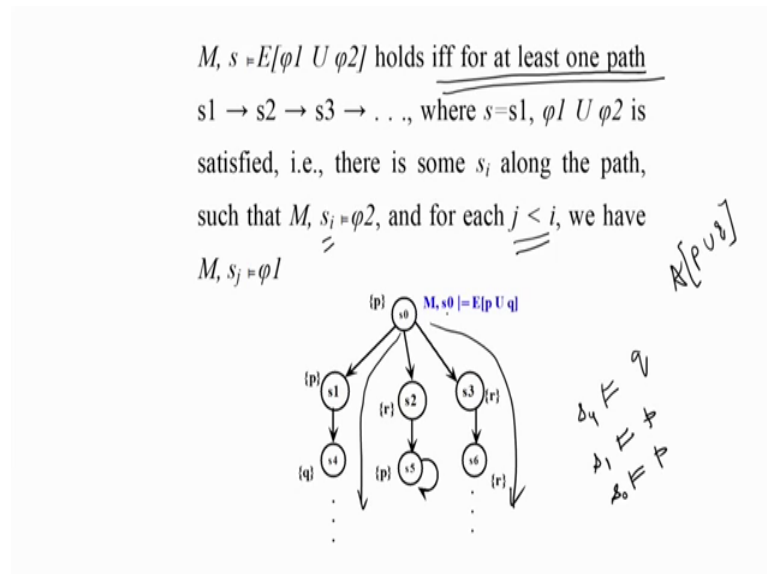
So, next operator is your until operator already I have mention that it is an binary operator. So, phi 1 a phi 1 until phi 2 again where in all path phi 1 remains true until phi

2 becomes true. So, that is why I says that you consider all possible such type of path s_1 to s_2 to s_3 like that. Where s_i is equal to s_2 we are going to consider this particular s_i , that means starting state is your s_1 in all such path ϕ_1 until ϕ_2 satisfied.

Now we are going to consider all the possible paths and impossible paths ϕ_1 until ϕ_2 satisfied. What does it means that means, we are going to get some s_i along the path. So, we have we are going to consider all possible path and along the path we are going to get some s_i , where ϕ_2 is true in this particular s_i and for each j less than i .

So, this is your i minus one i minus 2 like that, so all those particular j s_j it models ϕ_1 , that means ϕ_1 must be true in all those previous state. So, we have to get some s_i along the path such that $M, s_i \models \phi_2$ and for each j less than i we have $M, s_j \models \phi_1$ then we can say that $M, s \models \phi_1 \text{ until } \phi_2$.

(Refer Slide Time: 60:51)

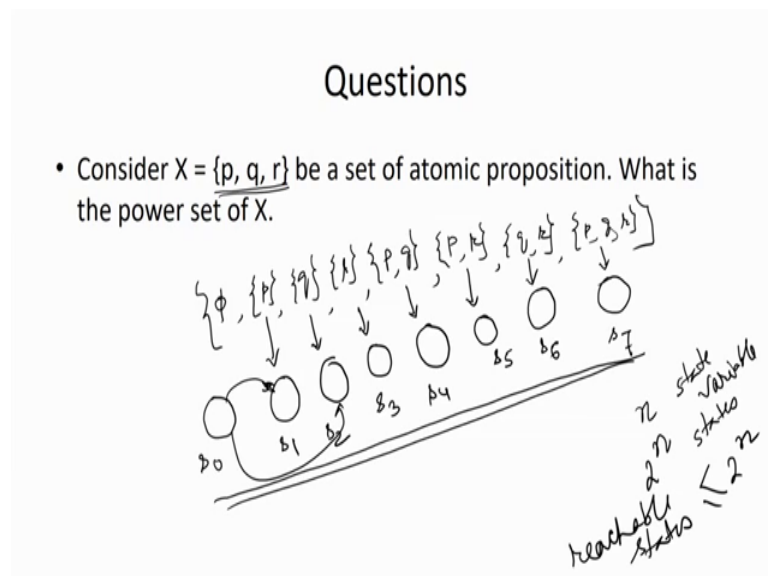


So, similarly we can define $E \phi_1$ until ϕ_2 the notion is same. So similar way we can define the formal semantics of this particular operator, only difference is here we are saying that if for at least one path. Now in case of a we are looking for all paths, but E we are looking for at least one path. Where ϕ_1 until ϕ_2 is satisfied in this particular path and what does it means we are going to get some state s_i , where ϕ_2 is true at this particular s_i and all its previous state j is less than i $M, s_j \models \phi_1$.

Now, here it is a simple example whether $E p \text{ until } q$ is true in this particular state s_0 . Now if I look into it then if you consider this particular path then what will happen, I am getting one state s_4 which models q and in this particular path what are the predecessor we are going to get s_1 which model p and s_0 which models p .

For all each j we have $M s_j \text{ model } \phi_1$, so you have seen that it models p its model p and only it models q then we are going to say that s_0 models $E p \text{ until } q$. But whether s_0 model $A p \text{ until } q$ if I look into it then we will find that it is not true, because at least in this particular path I am saying that p is not remains true and over q is also true may be some in future it may be q may be true. But p is not true in all the predecessor state, so $A p \text{ until } q$ is not true in this particular model at the state s_0 .

(Refer Slide Time: 62:54)



Now, these are the formal semantics of the CTL, so today what we have seen we have defined the formal syntax of CTL and formal semantics of the CTL and how to define the truth value of the CTL formula. How to construct correct CTL formula, how to get the semantics and CTL formulas or state formula and one basic notion just I have mention over here that every temporal operator must be preceded by a path quantifier.

Now, one simple questions now we are going to discuss it says that, consider the set X it is having 3 atomic proposition p, q, r ok. Now what is the power set of X I think you know the notion of power set. So, what are the power set of X if you see that that means,

generally if ϕ is to (Refer Time: 63:46) is a path subset or set of this particular power set.

I can say that p is one subset of this given set q is also a subset of this given set r is also a subset of this given set or I can say that $p \ q$ is a subset $p \ r$ is a subset and $q \ r$ is a subset also as per notation $p \ q \ r$ is also a subset of X . Do you find any other subset of this particular 3 element, if you inspect it you are not going to get any other subset, so this is the power set of this particular keep hand set.

Now, how many elements are there I think there are 8 elements because, we know the number of the power set is equal to 2^n , if n is the number of elements of a given set. Now, here what is this basic notion actually you just see if I look into the power set, basically it gives me all possible states of a given system. Why I am saying now we are talking about the atomic proposition and these atomic propositions are going to use to look for the state space system, what are the states we are having in Kripka structure. You just see that we are having the labeling function and it is states are labeled with the atomic proposition which are true in this particular state. So, whatever subset we have written, we said that these are the truth values of this atomic proposition is true in this particular subset.

Now, in this particular case I can say that this is the state s_0 , where none of the atomic proposition $p \ q \ r$ is true. I can say this is the state s_1 where the atomic proposition p is true, this is the state s_2 where the atomic proposition q is true. This is the state s_3 this is the atomic proposition r is true and s_4 is the state where the atomic proposition p and q is true s_5 is the state where the atomic proposition p and r are true and s_6 is the state where the atomic proposition q and r are true and s_7 is the atomic proposition where all the 3 atomic proposition $p \ q$ and r are true.

So, if you look into the combination of the truth values we are going to have these particular 8 different possibilities and these 8 possibilities can be now treated as the state of the system. If I am going to work with a system where it is having 3 variables, those 3 variables can be treated as my atomic proposition of the system and truth values of this atomic proposition will be either true or false and when we mapped it to the digital system, then we can say that this is either on or off or we are going to represent it with 0

and 1 in our digital logic system ok. Embedded system is also implemented with the help of digital logic system.

So, if we are having 3, if we are working with 3 atomic variables or such tree state variable then number of possible states are 8 only we cannot get more than 8. So, if we are working with n different variables or n different system variable, then we are going to get 2 to the power n states. So, we are going to have finite number of state whatever be that n maybe, if we are working with hundred system variable that system variable may be either on or off state, you are going to get 2 to the power 100 different states which is again finite in nature.

So, that is why in our temporal logic we are talking about the atomic proposition, when we work with our system digital system this atomic propositions are nothing but our system variables. Those system variable states may be either on or off on means true off means false. So, this is the mapping basically and now depending on the number of state variable we are going to have the state space and all the state space is finite. But when we are going to design a system all states may not be reachable some of the states configuration we may not achieve.

So, if I am having n state variable, then we are going to get 2 to the power n states ok. So, as for example I am having 3 state variable p q r or I can say atomic proposition, so these are the possible state behavior and we can have a transition from s_0 to s_1 or s_0 to s_2 depending on our system. Now, in some system all those particular states may not be a valid configuration. So, in that particular case we are going to say reachable states, where reachable state is always a subset of all those particular state which is either less than equal to 2 to the power n ok.

So, these are the valid configuration of the system, if some configurations are not valid then that will not come into the picture and we will say that this is not valid of our system ok. So, this is the scenario so total possible states is your 2 to the power n , but in our in reality reachable states will be less than this 2 to the power n because all configuration may not be possible because, it may have some conflict we will see with some example.

(Refer Slide Time: 70:12)

Questions

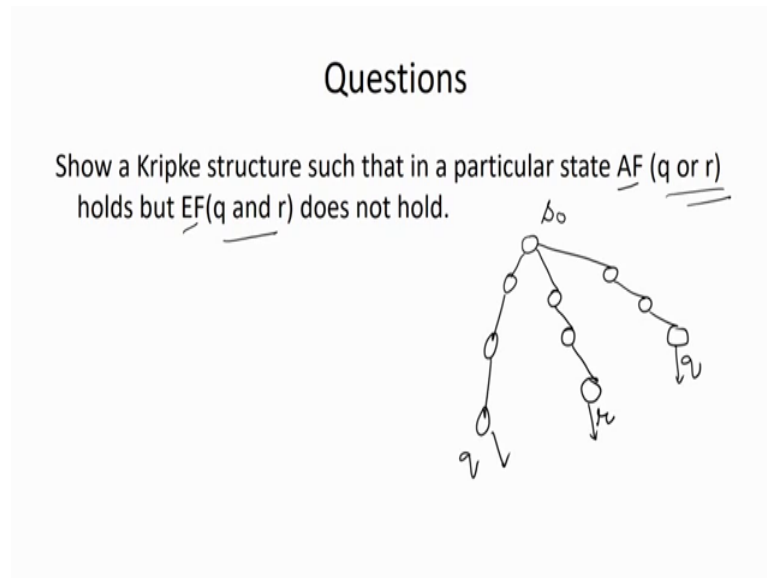
Show a Kripke structure such that in a particular state $EX(q \text{ or } r)$ holds but $EX(q \text{ and } r)$ does not hold.



Now, here said that another question related to our temporal logic show a Kripke structure, such that in a particular state $EX p \text{ or } q$ holds, but $EX q \text{ and } r$ does not hold. So, it says that basically I can give a some notation example over here. So, if I am going to say that whether in this particular state q or r is true or false.

So, in that particular case so at least I am having one state or in all state q or r any one of these things is true or I can say that ok. But whether $ex q \text{ and } r$ is true here q is true, but r is not true here r is true but q is not true, but here nothing is true. So, $EX q \text{ and } r$ is not true in s_0 , but $EX q \text{ or } r$ is true. So, this is $AF q \text{ or } r$ and $EF q \text{ or } r$, so one is in all path q and r and there exist a path q and r can you give a model.

(Refer Slide Time: 71:35)



So, this is some very simple construct I can say. In this particular case say if I go this particular, but in future I am getting at least q is true. So, q or r is true in this particular path so r is true. So, q or r is true in this particular path q is true, so q is r is true. So, in this particular state s_0 in all path in future q or r is true.

But whether there exist a path in future q and r holds. So, in this particular model if you find that in future q not going to get any state where q and r is true here q is true but r is not true, so q and r is false here also q and r is false here also q and r is false. So, we are not going to get any path where q and r is true in future ok. So, these are the way you can see that truth values.

(Refer Slide Time: 72:47)

Questions

Express the following property in CTL:

It is possible to get a state where started holds, but ready does not hold.

$EF(\text{started} \wedge \neg \text{ready})$

Now, some properties now we are CTL is used to express our properties, so if properties says like that it is possible to get a state where started holds but ready does not hold. So, basically now it is a system we are going to work say if we are switch on the system then we can said we have started a system, but we are having some other condition now whether system is ready or not ok. Sometimes you can look in to the printer you say that it is switch on, that means already you have started a printer. But it is not ready due to some other reasons maybe something has gone wrong do the system.

So, is it possible to get a state where started holds but ready does not hold. So, in a system we can reason such type of properties. So, when we are going to reason about such type of properties we have to express these properties with a formal notation. So, here we are talking about the CTL, so we will see how CTL is used to express this particular property.

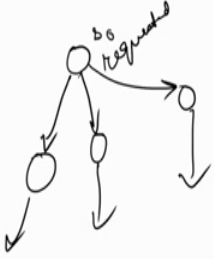
So, it is something like that started holds but ready does not hold. So, started and not of ready so is it possible to get a state. That means, in the system whether we are going to get such type of situation, so there exist a path in future. So, you are saying that whether we are going to get any computational path or any, so that in future started and not ready this is going to happen ok. So, this is the CTL representation of this particular property is it possible to get a state, where started holds but ready does not hold.

(Refer Slide Time: 74:34)

Questions

Express the following property in CTL:
For any state, if a request (of some resource) occurs, then it will eventually be acknowledged.

AG(requested → AF acknowledged)



Another one (Refer Time: 74:36) for any state if a request for some resource occurs then it will eventually be acknowledged. So, in system it happens with request for some resources if it is available then it will be granted eventually ok. So, for any state if a request for some resource occurs then it will eventually be acknowledged, now how we are going to represent this particular property formally in CTL.

So, if in a particular state if request is hold it is request for some resources, then wherever you go in all path in future we are going to get acknowledged. So, say in this particular state we are requested or some resources ok, so requested is a atomic proposition one we request then this truth values of request. So, from this particular s 0 we may have different possible transition depending on the system behavior, so it says that wherever you can go in future so that means in all path in future somewhere we should get the acknowledgement ok. Because, we have requested for our some resources now system is having several behavior transition behavior, so wherever we proceed finally, we should get the acknowledged.

So, requested implies in all path in future acknowledged and it says for any state. So, when we design a system such type of requirement may be true or may be required in all possible states. So, that is why it says that all path globally this must be satisfied ok. So, this is the one system behavior one system property, now when we design the system then we have to check such type of properties or behavior. So, formally we have to

capture it and we know the meaning now we will see how we are going to check those particular properties.

So, you just see that if we are having some system behavior that can be captured with the help of CTL formulas and those CTLs formulas will be used to give the specification and we are going to check whether those particular specifications is true in our model. Model is nothing but my design of the system ok. So, this is the way we can think how CTL will be used while designing the embedded system, with this I am coming to the end of today's lecture.

Thank you all.