Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 47
Final Algorithm

We are in unit 10 of this course. We are talking about algorithms for Computation Tree Logic. In module 1 and module 2, we have described the prerequisites for giving the final algorithm. Let us see the algorithm in this module.

(Refer Slide Time: 00:24)

CTL model-checking problem

Given transition system M and a CTL formula ϕ, find all states of M that satisfy ϕ

- Module 1: Every CTL formula can be written using **EX, EU, EG**
- Module 2: Labelling algorithms for **EX, EU, EG**

We were looking at this problem given a transition system M and a CTL formula phi, find all states of M that satisfy phi. This is known as the CTL model checking problem. In module 1, we saw that every CTL formula can be written using EX, EU and EG. It was called the existential normal form. In module 2, we have given algorithms for model checking, CTL formulas of this form, EX, EU or EG.

(Refer Slide Time: 01:05)

Coming next: Generic algorithm for a CTL formula

State formulae

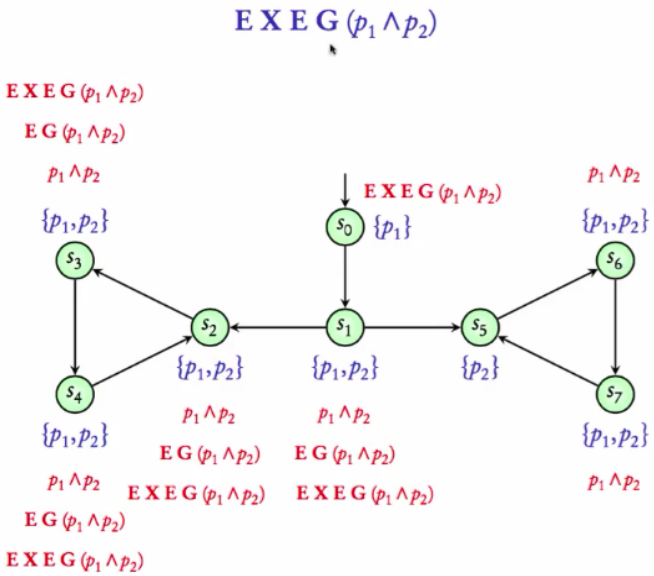
$$\phi := \text{true} \mid p_i \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid EX\phi \mid E(\phi_1 U \phi_2) \mid EG\phi$$

$p_i \in AP$ ϕ, ϕ_1, ϕ_2 : State formulae

Now we will look at a generic algorithm for a CTL formula in existential normal form. So in module 2, it was considering EX of some simple atomic formula like, P1, P1 and P2, P1 and not P2 and so on. However, the existential normal form also allows formulae like EX, EG, E of EG P1 until EG P2 things like this, so given a generic formula in CTL ENF, how would you model check it.

In other words, how would you find the states of the transition system that satisfy this formula.

(Refer Slide Time: 01:55)



Let us start with an example. So this is a transition system and we have been given this CTL formula in ENF, EX, EG, P1 and P2. The first task would be to look at this inner most sub-formula P1 and P2. Let us first label the states that satisfy P1 and P2. These are the state that

satisfy P1 and P2, s5 and s not do not satisfy this. Let us now look at EG P1 and P2. What was the algorithm for this? The first step labelled every state with EG P1 and P2.

In the next step, we removed EG P1 and P2 in all states where P1 and P2 is not true. So look at the states, s not does not satisfy P1 and P2, s5 does not satisfy P1 and P2. So, first remove the label EG of P1 and P2 from these two states. Now according to the algorithm, we keep looking at states and if every successor of that state does not have a label EG P1 and P2, then we should remove the label from this state as well. So look at S7.

The only successor is here and it does not have the label EG P1 and P2, so we remove it from here. Look at s6, for the same reason, we remove the label from here as well. What about s1? s1 has a successor, which is labelled EG of P1 and P2, so do not do anything. What about s2? It has a successor s3, which has this label so we do not modify this, then what about s4? It has a successor, which is labelled with EG P1 and P2, so we do not do anything to this as well.

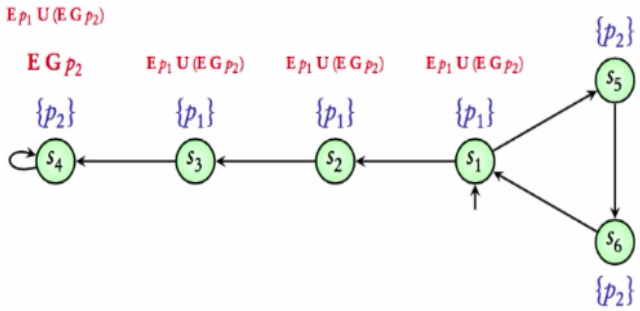
So if you look at this state, there exists a path where P1 and P2 is true always, same for this state, this state and this state, okay. So we are done with EG P1 and P2. Let us now look at the final formula EX of this. We know that if we have already labelled the states where this inner formula is true, when we have an algorithm to find the states where EX of this inner formula is true for this inner formula phi. So phi is labelled here, here, and here.

Then what was the algorithm for EX doing? It looked at some state and will see if there exists a transition to a state where phi is true, then this state would be labelled with EX of phi, right. In that way, this state has a transition to a state that satisfies EG P1 and P2 so EX EG P1 and P2 is true here. Similarly look at this state. It has a transition to a state that satisfies EG P1 and P2, so this state satisfies EX EG P1 and P2. Similar for this, this, and this.

So all these satisfy, there exists a path such that the next state satisfies EG of P1 and P2. Essentially, we were doing this labelling algorithm hierarchically. We started with the smallest sub-formula, then we looked at this bigger formula and then we finally looked at the entire formula, but by the time when we looked at the entire formula we have already marked the states where the inner formula is true, then we made use of existing algorithm for EX.

(Refer Slide Time: 06:18)

$$E\,p_1\,U\,(E\,G\,p_2)$$



Let us look at another example. Now we want to check if E of P1 until EG P2 is true. So there is a bracket here. E of P1 until EG P2. Let us first look at EG P2 and mark the states of this transition system that satisfy EG P2, so the algorithm, how does it work? It first labels all states with EG P2 and then the first step removes the label EG P2 from all states that do not have P2, so they are this, this, and this, okay.

Now look at predecessors. Look at the predecessor of s1. All successors of this state do not satisfy EG P2, in the sense that the label has been removed. So we have to remove the label from here. Similarly, s5, all successors do not satisfy EG P2, so remove the label from here, the only state that remains is this, look at its successor. Its successor is itself so we cannot remove EG P2 from this state, fine.

Now let us look at E of P1 until EG P2. We had looked at an algorithm for E of phi 1 until phi 2. Here phi 2 has been labelled. We also know the labels for phi 1. Let us now apply the algorithm for E of phi 1 until phi 2, okay, so this is the only state, which satisfies EG P2. As a first step, mark E of P1 until EG P2 to be true here. Look at its predecessors, s3 is a predecessor. It satisfies P1, so this state also satisfies P1 until EG P2.

There exists a path that satisfies P1 until EG P2. Look at its predecessor, it satisfies P1, so you need to label this state with E of P1 until EG P2 as well. Similarly look at this predecessor, it satisfies P1, so you mark this state with E of P1 until EG P2. Look at s1's predecessor. This does not satisfy P1, so we do not change the status of this state. We do not

mark EG P of P1 until EG P2 for this state and then the algorithm terminates, because there is no change.

(Refer Slide Time: 09:22)

```
function SAT(φ)
  /* Input: Transition system M with state set S, CTL formula φ in ENF */
  /* Output: Set of states satisfying φ */

case
      φ is true : return S
      φ is p_i : return {states containing p_i }
      φ is φ_1 ∧ φ_2 : return SAT(φ_1) ∩ SAT(φ_2)
      φ is ¬φ_1 : return S − SAT(φ_1)
      φ is E X φ_1 : return SAT_EX(φ_1) /* procedure seen in Module 2 */
      φ is E (φ_1 U φ_2) : return SAT_EU(φ_1, φ_2) /* procedure seen in Module 2 */
      φ is E G φ_1 : return SAT_EG(φ_1) /* procedure seen in Module 2 */
end case
end function
```

Let us now write the final algorithm. The input is a transition system with the set of states s, so it would be a transition system like this with some set of states. We denote the set of states with s along with it a CTL formula in ENF is given. So here, you have this as your input and the CTL formula in ENF. The output of this function should be the set of states, which satisfy the CTL formula. So if you give this transition system, it should output s1, s2, s3, and s4.

Let us see, so this would be a recursive procedure called SAT. There are cases, if the formula in consideration is just true, then all the states satisfy true. So the function should be written s. If phi is Pi, that is it is something like P1 or just P2, then the function should written the set of states that contain Pi. For example, if the formula was just P1, then the function should written s1, s2, s3.

If phi is phi 1 and phi 2, then run the function on phi 1, run the function on phi 2, and then take the intersection. SAT of phi 1 will give the set of states that satisfy phi 1, SAT of phi 2 will give the set of states that satisfy phi 2, and now we need the intersection of these two. See, if you see we are building up using the syntax of ENF. What was the next one? Phi is not of phi 1. If phi is of this form, then run SAT on phi 1.

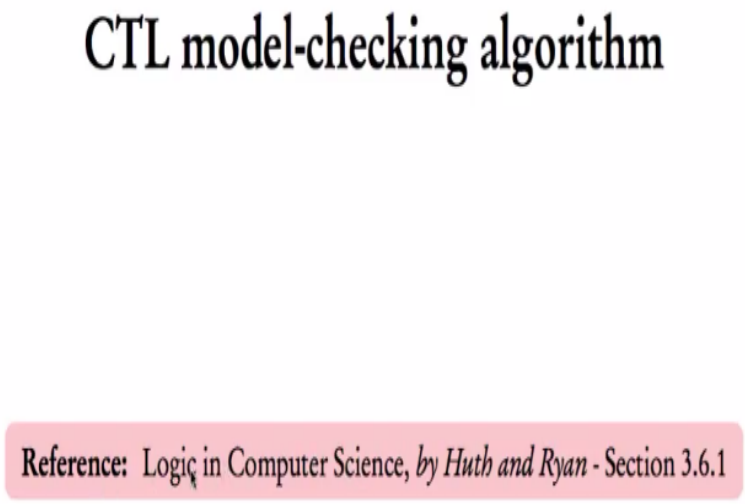
All states that remain, in the sense, all states that do not satisfy phi 1 is what we want, so written s minus SAT of phi, s is the set of states, SAT of phi 1 is the set of states satisfying phi

1. Now comes EX. If phi is of the form EX of phi 1, then run the procedure that we saw in module 2, you first label the states where phi 1 is true and then mark the states where there exists a successor labelled phi 1. We denote this algorithm by SAT EX of phi 1.

If phi is E of phi 1 until phi 2, then label phi 1, label phi 2, that is run SAT on phi 1, run SAT on phi 2, and then recall the algorithm for E of phi 1 until phi 2 as seen in module 2. This procedure is denoted using SAT EU of phi 1, phi 2. Finally, if phi is EG phi 1, then run SAT over phi 1 and then use the procedure that we saw in module 2, where you first label all states with EG phi 1 and then keep removing repeatedly until there is no change.

This procedure is denoted as SAT EG phi. This is the final algorithm. Given a CTL formula and a transition system, this function returns the set of states of the transition system, which satisfy a CTL formula.

(Refer Slide Time: 13:48)



So the reference to this algorithm is the book Logic in Computer Science by Huth and Ryan. This is in section 3.6.1. In this book, you can also get the pseudocode for SAT EX, SAT EU, and SAT EG. This completes the discussion about the algorithm for CTL.