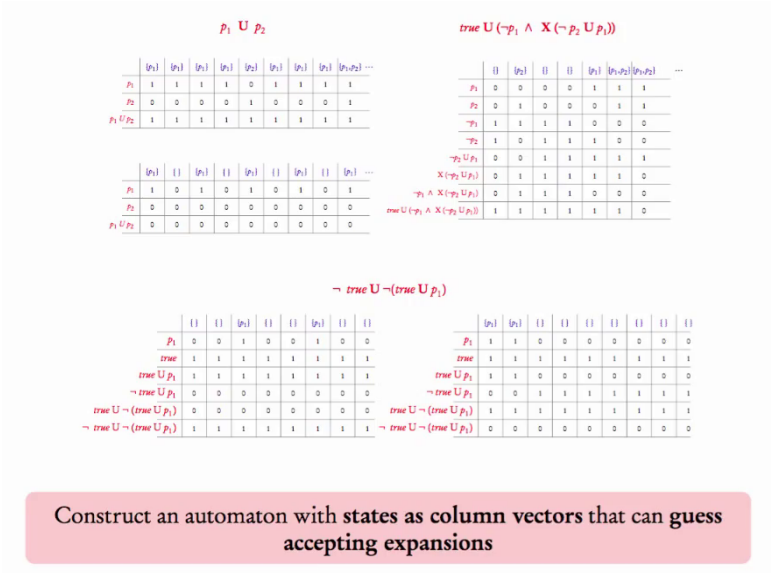
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology- Madras

Lecture – 39
Automaton Construction

In this module we will see the final algorithm for converting an LTL formula into a Buchi Automaton.

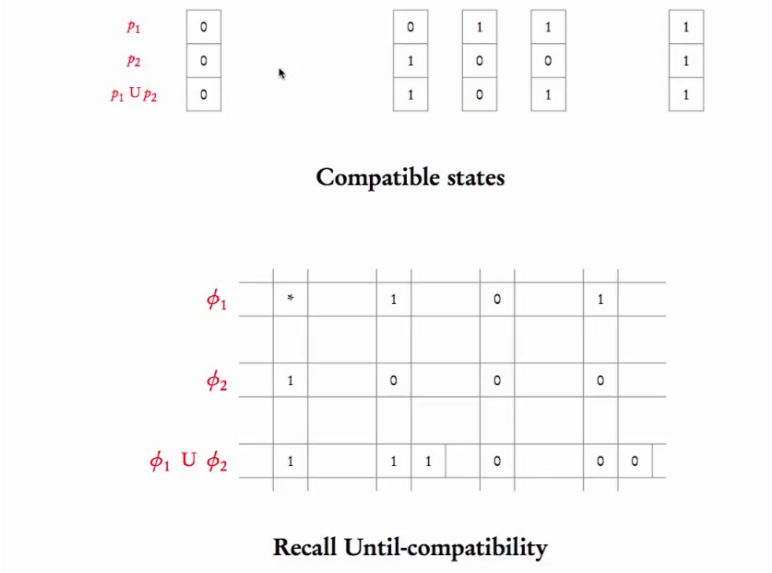
(Refer Slide Time: 00:11)



In the last module we saw what are called formula expansions given an LTL formula and a word in the formula expansion we evaluate every sub formula at every position in the word this gives us some kind of an infinite table. We also saw the properties satisfied by this formula expansion, for example if X of some formula is 1 here then the next position that formula should be 1 similarly we saw AND-NOT Until compatibility conditions.

We now want to construct an automaton whose states are these columns and the transitions are given in such a way so that on a word which belongs to the language of this formula the run of the automaton looks like the expansion we want the automaton to guess accepting expansions. We will see an example and then we will give you the final construction.

(Refer Slide Time: 01:39)



Let us start with the example p 1 U p 2 what are the possible columns in the expansion of p 1 U p 2 the sub formulae are p 1, p 2 and p 1 U p 2 and each of them could be either 0 or 1 so in all we have eight possibilities for a column in the expansion, however recall that there are some compatibility conditions in particular look at the Until compatibility condition it says that if phi 1, phi 2 and phi 1 U phi 2 are there has sub formula - has sub formulae then these are the possible values in the columns.

Let us see what about 000000 is a possibility it says that phi 1 is 0, phi 2 is 0 and phi 1 U phi 2 is 0 as well what about 001, 001 is not possible because this says that p 1 is 0 op 2 is 0 but p 1 U p 2 is true in that position this cannot be the case, so this cannot be a potential column in the expansion so we remove that what about 010 this says that p 2 is true if you look at the compatibility criterion if phi 2 is true then phi 1 U phi 2 has to be 1.

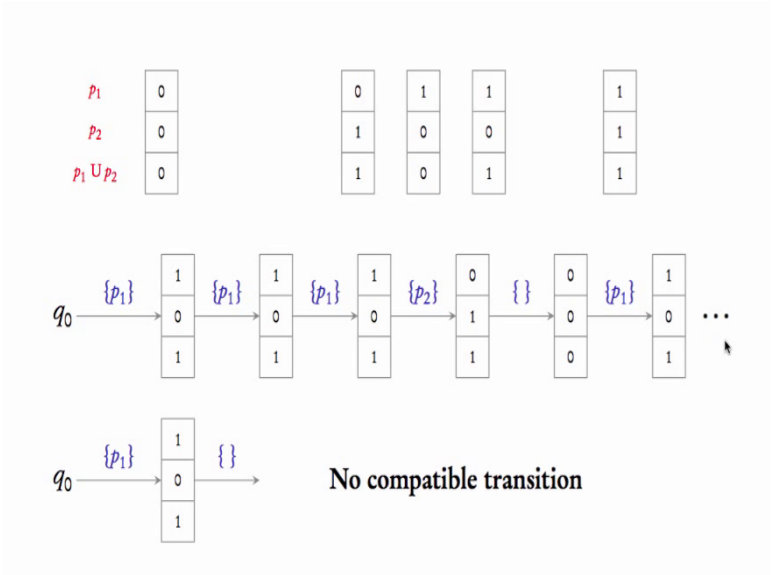
And here p 2 is 1 but p 1 U p 2 is 0 this is not possible so we remove that vector as well so that cannot be a possible state, what about 011 this is possible when you have 1 here and 1 here either this is 0 or 1 so this is possible what about 100 it says that 100 is potentially possible but if this is the case then in the next step the phi 1 U phi 2 should be 0 that is fine we will take care of the next step later.

But for now 100 is a possible column so we keep that, 101 is again possible again if 101 occurs as a column then there is a condition on the next step we will not bother about it now we know that 101 can potentially occur as a column so we keep 101 what about 110 if phi 2

is 1 then phi 1 U phi 2 should be 1 so this cannot be the case I mean this is not a compatible column so we remove it 111 is possible.

So these are the blocks which can potentially occur in an expansion of p 1 U p 2 you can go back and check the expansion of p 1 U p 2 the example that we had given every column would be one of these. These are called compatible states we will be constructing an automaton using states given by this columns.

(Refer Slide Time: 05:11)



Before giving the actual construction let us try to see what the automaton should do on a word let's assume that there is a separate initial state q0 and this automaton should accept words which satisfy p 1 U p 2 and how are we going to do that we are going to guess the accepting expansion for the word.

Suppose the first letter that is read is p 1 how does the first column look like p 1 should be 1 right so it could potentially be this one or this one since p 2 is 0, it can be one of these two states since we are guessing and accepting expansion the automaton guesses that the rest of the word is going to satisfy p 1 U p 2 and so from q0 the automaton should go to 101 the automaton is making a guess that the rest of the word will satisfy p 1 U p 2 if it does not satisfy p 1 U p 2 automaton should be blocked.

Let us see so now we are in state 101 suppose the next letter that is read is p 1 again if you look at the column the evaluation would have been 1 because of p 1, 0 because there is no p 2 and now by the criterion by the Until compatibility criterion if you have 101 hear the next

step should have 1 here so the automaton will choose the state 101 this says that p 1 U p 2 is going to be true and currently p 1 is true so right now only p 1 is true so still p 1 U p 2 is going to be true that is what the automaton has guessed.

Now if it still sees p 1 it will be keep staying in this state now what if is it p 2 because it is sees p 2 this entry is 0 this is the entry for p 1 this is 0 this is going to be 1 and of course this is going to be 1 because if this is 1 then this has to be 1 just look at the word p 1 p 1 p 1 p 2 and try to build the expansion for this word you will see that this will be the first column this will be the second column this will be the third column this will be the fourth column.

So the automaton had guessed that at some point it will come and it has now seen p 2 and it has verified this guess. Suppose it sees an empty letter I mean the letter consisting of the empty set this and this have to be 0 here there is no choice it has to be 0 because p 1 U p 2 cannot be true in this position as p 1 is false and p 2 is false, now at p 1 suppose it sees p 1 this has to be 1.

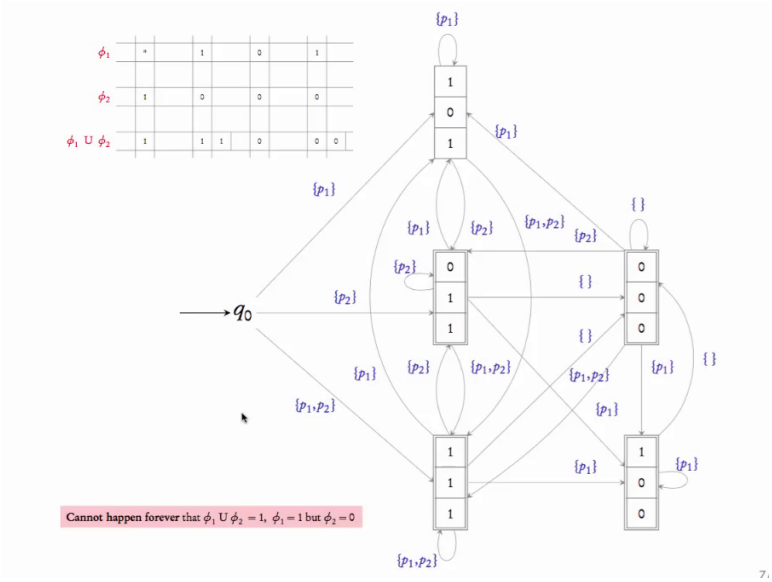
This has to be 0 and here it can make a guess it can guess whether in the – in the rest of the word p 1 U p 2 is going to be true or not here we have assumed that it is 1 and then now the automaton has to verify if p 1 U p 2 is true. Let us see one more example of a word initial state is q0 on seeing p 1 the automaton has two choices it can go to 100 or 101 however we enforce the automaton to go to 101.

Because we want to recognize accepting expansions only so the automaton makes a guess that p 1 U p 2 is going to be true now. Suppose it sees the empty letter it knows that p 1 U p 2 is going to be true that means in the next step either p 1 should be or true p 2 should be true so if it sees this it should not be able to take the next transition so from this state there should be no transition on this empty letter there should be no compatible transition that reads this letter.

So this is the rough idea the states of the automaton are going to be these phi along with q0 initially on reading some letter the automaton goes to the corresponding state with the last entry being 1 it assumes that in the rest of the word p 1 U p 2 is going to be true and based on the compatibility criteria the next transitions should be defined.

In particular if the automaton reads p 1 followed by the empty letter it should not be able to move however if it reads p 1 p 1 p 1 p 2 it should come up to this state and here it should go to this state and here it should go to this state etc.

(Refer Slide Time: 10:54)



Let us now give the automaton construction the states are q0 and these columns these were the compatible states this is the Until criterion we should give a transitions based on this Until criterion this says that if 101 is here then there is some condition on the next step similarly if there is 100 here there is some condition on the next step, first from q0 what should the transitions be the transitions should go to states where the last entry is 1.

Because in accepting expansions the last row of the first column is 1 we want our automaton only to look at accepting expansions, so from q0 if it is sees just p 1 it goes to 10 this is because of the word compatibility criterion if you have p 1 then the p 1 entry should be 1 and the p 2 entry should be 0 and of course the last entry we wanted to be 1 if it is p 2 then it goes to 011 if it is p 1, p 2 it goes to 111.

If you look at any accepting expansion that starts with p 1 p 2 then this will be the first column similarly if you look at any accepting expansion that starts with p 2 this will be the first column and similar for this thing if the accepting expansion starts with the word p 1 than this will be the column. Now let us look at this state what are the outgoing transitions for this state?

If you look at the Until compatibility criterion it says that if 101 occurs then in the next step this entry has to be 1 so from this state you can either stay in this state or go here or go here you cannot move to the states, that is what is depicted here. If you see p 1 if the next letter that is read is p 1 then you stay in this state if the next letter that is read is p 2 you jump to this state if the next letter that is read is p 1 p 2 the automaton jumps to this state.

Because p 1 and p 2 both are true and the Until compatibility criterion asks that the last letter has to be 1. Let us now look at this state when you have 11 here there is no condition on the next step the next step could be any of the possibilities we just need to ensure word compatibility let us see if you read the empty letter there is only one state which can - which can be reached this is 0 0 and 0 what about letter p 1.

You can either go to 101 or 100 automaton is - if it goes to this state the automaton is making a guess that the less - rest of the word satisfies p 1 U p 2. However if the automaton used to this state then it is making the guess that the rest of the word does not satisfy p 1 U p 2 here there is non determinism, here there are two choices on the letter p 1 automaton can either go here or it can go here now if it reads p 2 it stays here if it reads p 1 p 2 it goes here that is depicted here.

Let us look at this state similar to this state there is no condition on the next step so when you read the empty letter you go here when you read p 1 there is a choice either you can go here or you can go here if you read p 2 you can go here and if you read p 1 p 2 you stay here there this transitions. Now let us look at this state and let us try to give the outgoing transition from this state.

Look at the compatibility criterion from 000 there is no condition on the next step so if you read empty you stay here if you read p 1 you go here or here if you read p 2you have to go here and if you read p 1 p 2 the automaton goes to this state. Now finally let us come to this state here there is a compatibility criterion for the next step it says that if the current vector is 100 the current column is 100 the next step has to be 0 in this entry.

This says that if p 1 is true here but if p 1 U p 2 is false here and p 2 is false here it cannot be the case that p 1 U p 2 becomes true here, see if p 2 is already false here the only hope of satisfying p 1 U p 2 is to have some p 1 here and then finally p 2, now if this entry says that p

1 U p 2 is true then this whole - this also has to be true, so in particular if you have 100 in the next step you should see only 0.

So from here there are only two choices either you can go here are you can stay here so this you can go if you read the empty letter and you stay here if you read the letter p 1 so we have given the states and transitions of the automaton so there are paths in this automaton the path - one path is unique p 2 and if you keep the reading p 2 then the columns are going to be like this.

So what you can check out is gives some infinite word and then try to find the runs of this automaton on that infinite word you will see that if that word satisfies the LTL formula then there will be a way of reading the entire word, however if the word does not satisfy the LTL formulas then this automaton will not be able to read after some point of time.

For example, the word p 1 followed by the empty letter does not satisfy p 1 U p 2 so you see p 1 empty letter empty letter empty letter and so on from q0 on p 1 you go here and here there is no transition out of the empty letter we had seen this example previously suppose you have the word p 2 followed by the empty letters then that is possible because you read p 2 and then you go to the state on reading the empty letter.

And then you can keep reading the empty letter any number of times the moment you read p 2 you know that the word satisfies p 1 U p 2. There is still one condition left, note that we have not mentioned what are the accepting states in this automaton recall the Until eventuality condition, for example look at this path p 1 p 1 p 1 p 1 p 1 and so on the word which consists only of p 1 can be read by this automaton.

But it should not be accepted because you are never seeing p 2, the Until eventualities condition will not be satisfied. Note that the condition says that if it cannot happen forever that phi 1 U phi 2 is 1, phi 1 is 1 but phi 2 is 0, now how do we eliminate words of this form here we will make use of accepting states we will mark the rest of the states as accepting the states where either phi 1 U phi 2 is 0 or the states where phi 2 is 1.

You can safely loop in the states you can see this states infinitely often and there is not a problem. However, if you loop in this state forever without visiting any of these things then

that word should not be accepted, now look at this word p 2 followed by empty letter empty letter and so on that is fine p 1 p 2 followed by p 1 followed by p 1 is fine because you have already satisfied p 1 U p 2 and then if you keep reading p 1 that is not a problem let me repeat so up to this point we added compatible states and we added transition that satisfy the compatibility criterion.

However we have by doing this construction we have still not satisfied the eventuality condition, for example this word p 1 p 1 p 1 p 1 and so on, this is a word which is not in the language and this is not an accepting expansion because it does not satisfy the eventuality criterion, from this construction we need to filter out only those runs which mimic accepting expansion.

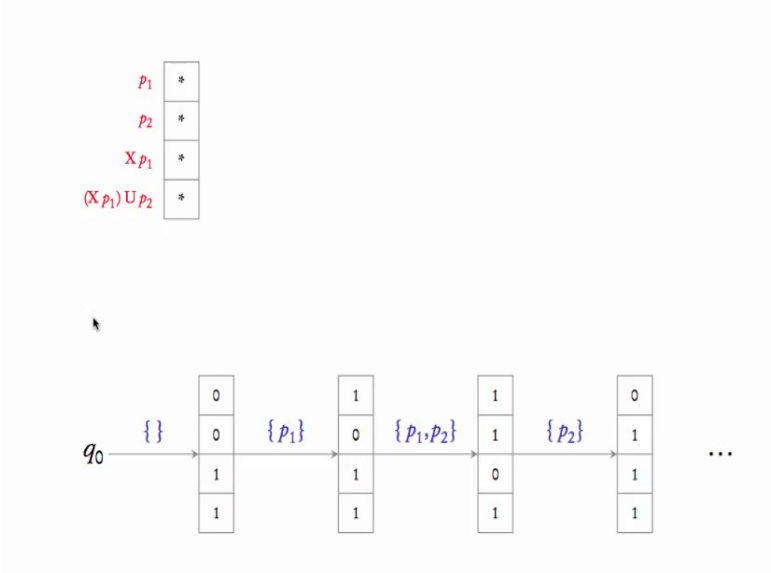
To do that recall the eventuality criterion it says that it cannot happen forever that you see a state where phi 1 U phi 2 is 1, phi 1 is one but phi 2 is 0. To ensure this criterion we will make this as a non-accepting state and states where phi 1 U phi 2 is 1 and phi 2 is 1 are safe and stage your phi 1 U phi 2 is 0 they are also safe.

Only the states where phi 1 U phi 2 is 1 phi 1 is 1 and phi 2 is 0 these are the states which cannot keep occurring forever that is continuously they cannot occur so apart from those states you make the rest of the states as accepting states, now look at any run in this automaton any accepting run in this automaton that will correspond to a word which satisfies p 1 U p 2 this is the automaton corresponding to the LTL formula p 1 U p 2.

Note that we had given an easier automaton for p 1 U p 2, however in module 1 we were not giving a mechanical way of constructing the automaton it was not an algorithm we were making use of our intuitions to create different automata for different LTL formula now this will be a standardized algorithm for any arbitrary LTL formula and that algorithm will give us this automaton for p 1 U p 2.

Let us see one more example we will not construct the entire automaton for Xp 1 U p 2 but we will see the steps towards constructing the automaton.

(Refer Slide Time: 22:55)



Firstly, the sub formulae or p 1 p 2 Xp 1 (Xp 1) U p 2 so if you had written an expansion for this formula you would have seen these as your columns, so this can take either 0 or 1, 0 or1, 0 or1, 0 or 1 so in principle there are 16 such states - 16 such possible values for columns however some of them would be incompatible.

Let us see some potential runs of the automaton for (Xp 1) U p 2 it starts with q0 if it sees the empty letter it has to go to a state where the first two entries are 0 this slot is for (Xp 1) at this position the automaton can only make a guess whether in the next step p 1 will be seen or not suppose the automaton guesses that Xp 1 is 1.

Let us see what happens but before that what about this lot in and accepting expansion we will always need that to be 1 okay, from q0 we let the automaton guess that (Xp 1) U p 2 is going to be true and so Xp 1 also needs to be true. Now since we have made that Xp 1 is true here it has to read a letter that contains p 1 and it goes to a state where p 1 is 1 p 2 is 0 it still has to maintain Xp 1 to be 1 because we have to maintain that (Xp 1) U p 2 should be true.

So the automaton has guess that (Xp 1) U p 2 is true, so this entry has to be 1 till its sees p 2 so Xp 1 has to be 1 if you recall the Until criterion this is our phi 1 this is our phi 2 if phi 1 U phi 2 is 1 and phi 1 is 1 and phi 2 is 0 then in the next step phi 1 U phi 2 should be 1 okay.

So what about the next step suppose here the automaton sees p 1 p 2 firstly, since Xp 1 is 1 the automaton has to see a letter that contains p 1 p 2 so from this state there cannot be transitions out of the empty letter and the letter consisting only of p 2 so it has to have a

transition either on p 1 or on p 1 p 2 now where does it go this and this have to be 1 and since p 2 is true any compatible state where p 2 is true will have (Xp 1) U p 2 to be 1 okay.

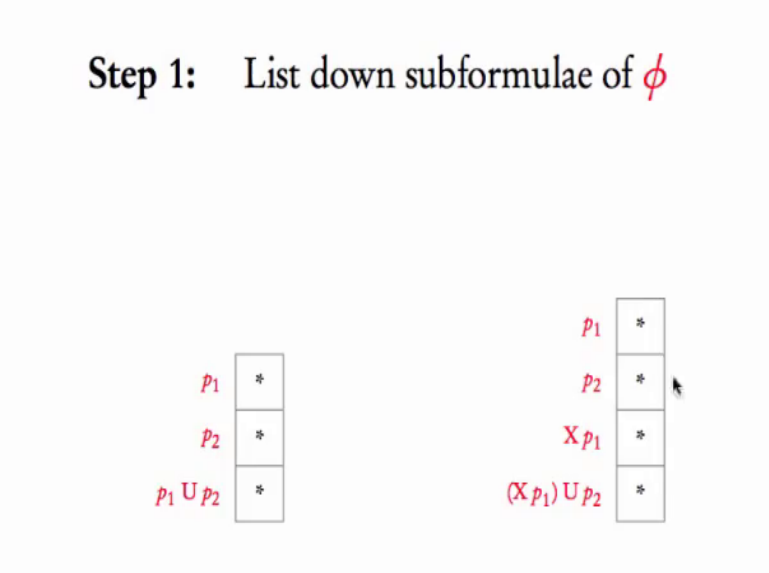
Now what about this position here this says that this is a guess about the next letter so here you can either say 0 or 1 and we have assumed to 0, since we have assume this to be 0 the next transition should not contain p 1 it should be either the empty letter or the letter consisting only of p 2 and here it has to be 1 here this has to be 0 here and here it has to be 1 because p 2 is 1 (Xp 1) U p 2 should be 1.

And here this could have been either 0 or 1 here the automaton has just 1 and so on. The idea is same you look at the sub formulae remove all incompatible states you look at all the compatibility criteria Until AND-NOT and then remove incompatible states some compatibility criteria like the X and Until have a condition for the next step and this can be ensured by defining the transitions appropriately.

For example, you here you have guessed that Xp 1 is 1 then from this state, the outgoing transitions will have to contain p 1 here if you have guessed that Xp 1 is 0 then the outgoing transition should not contain p 1. So this is the idea of instruction for (Xp 1) U p 2. we have seen the algorithm for p 1 U p 2 and we have given an idea for Xp 1 and p 2, in the case of p 1 U p 2.

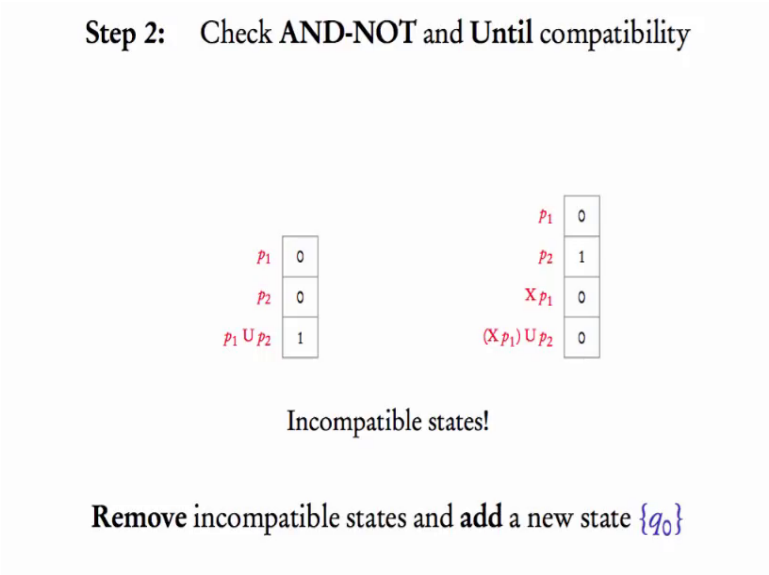
We also saw the final automaton that is constructed, in (Xp 1) U p 2 we gave some ideas as to what the states and transitions should look like. Let us now sketch the Construction for an arbitrary LTL formula phi.

(Refer Slide Time: 28:20)



Step 1 list down the sub formulae of phi you can go back to the slide with expansions and you can see the first step in the expansion was to list down the sub formulae in the case of p 1 U p 2 you had p 1 p 2 p 1 U p 2 as a sub formulae and this this gave rise to many states here in principle there could be 8 states for (Xp 1) U p 2 there could be in principle 16 states.

(Refer Slide Time: 28:55)



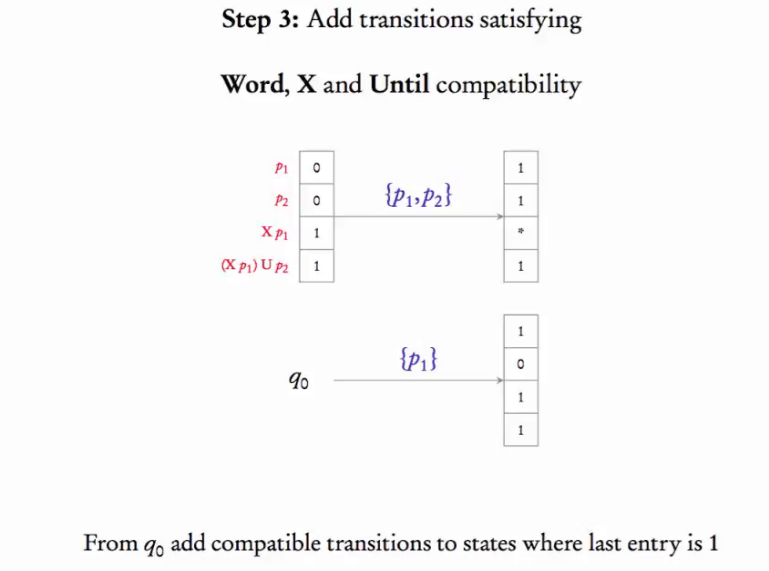
Step 2 on these states check the AND-NOT and Until compatibility, go and look at this AND-NOT compatibility this says that if the entry for phi 1 the entry for phi 2 and the entry for phi 1 and phi 2 should be compatible with respect to AND, so if phi 1, phi 2, phi 1 and phi 2 are sub formula of phi then the values in the columns should be compatible similar for Until.

For example p 1 0 p 2 0 but p 1 U p 2 be 1 is incompatible, similarly p 1 0 p 2 1 but (Xp 1) U p 2 0 is incompatible so based on this compatibility criterion remove all incompatible

states and add this new state q0 which will be our single initial state okay. Step 1 you listed down the sub formulae and made states which - which look like columns now from these columns remove incompatible ones, you will now get a set of compatible states.

These are going to be the states of your automaton and to this set of states add this new state q0.

(Refer Slide Time: 30:27)



Next step to define an automaton you need to give the states the transitions and the set of accepting states, so far we have given what are the states what about the transitions add your transitions so that the word compatibility X compatibility and Until compatibility are satisfied, for example in this state Xp 1 is 1 so the letters out of this state should contain p 1 if it is p 1 p 2 then you should go to the state where both p 1 and p 2 are 1.

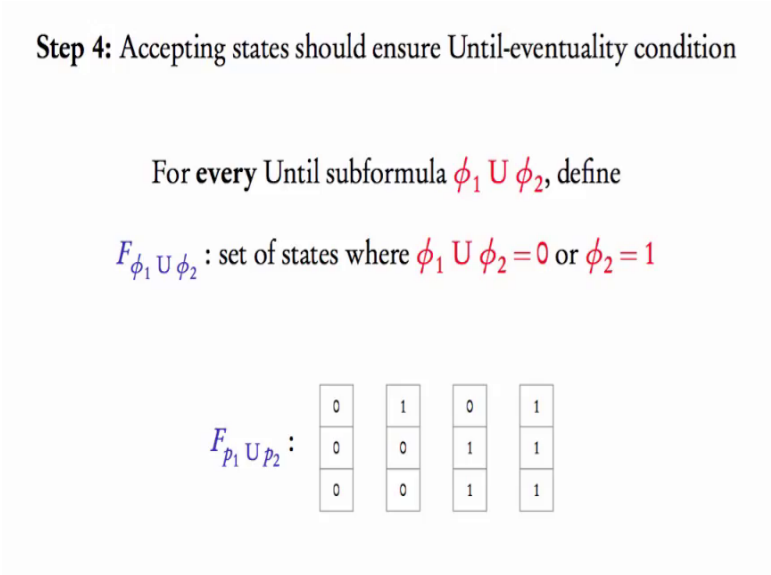
And any compatible state where p 2 is 1 would have this to be 1 this ensures word compatibility. This ensures the fact that here there is Xp 1 and then you are reading p 1 p2 that is it contains p 1 ensures X compatibility and here this is for the next step so this could be any value you could either make this to be 0 or 1 so on p 1 p 2 there are two transitions either you go to 1101 or 1111.

So the transitions can be added in such a way so that the word X and Until compatibility criteria are satisfied. From q0 which is going to be the initial state at compatible transitions as it add word compatible transitions in which the last entry is 1, similarly here this could have

been either 0 or 1 however the last entry has to be 1, from q0 you always go to a state where the last entry is 1.

This is for satisfying this condition about accepting expansions if you look at any accepting expansion the first column last row will be 1 so this is step 3. From Step 1, 2 and 3 what we have done is we have specified what the states are and we have specified what the transitions are.

(Refer Slide Time: 32:54)



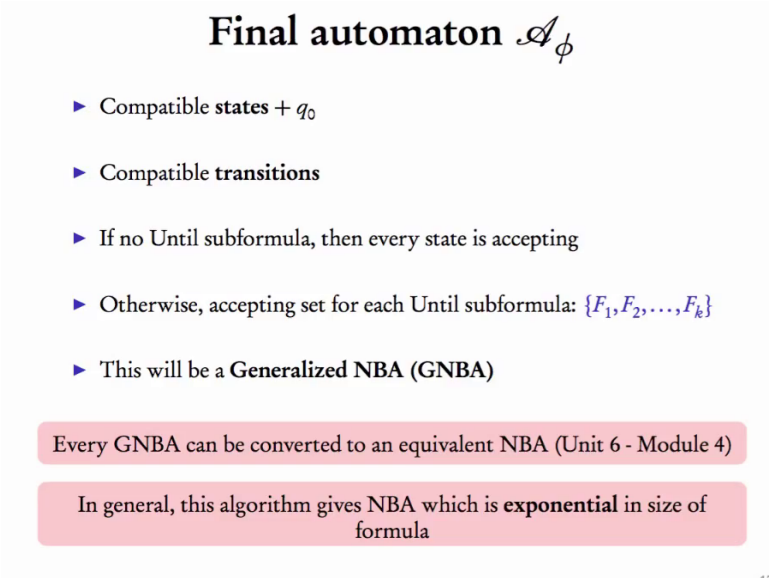
We still need to ensure the Until-eventuality condition as you saw in the p 1 U p 2 example we need to define accepting states in such a way so that it is not true always that phi 1 is true phi 1 U phi 2 is true but phi 2 is never true you should never reach opposition from where you always see this particular state so and there could be many Until formulas I mean there could be many Until sub formulas.

For example in the sub formula p 1 U p 2 and p3 U p4 for example if there are many automatons propositions or something like p 1 U p 2 and (Xp 1) U Xp 2 things like this there could be many Until formulae in 1 LTL formula for each Until you need to ensure the eventuality criterion, so what you do for every Until sub formula phi 1 U phi 2 define the set of states F phi 1 U phi 2 as the set where either phi 1 U phi 2 is 0 or phi 2 is 1.

Let me explain this with example of p 1 U p 2, p 1 U p 2 was a sub formula F p 1 U p 2 will be the set of states where either phi 1 U phi 2 is 0 in which case it is fine or phi 2 is 1 so the state that we have eliminated will have phi 1 U phi 2 to be 1 and phi 2 to be 0 so such states

are eliminated from the set of accepting states so for every Until sub formula you define F p 1 U p 2 to be the set of states where phi 1 U phi to is 0 or phi 2 is 1 this will be the accepting set for phi 1 U phi 2, any run that visits one of the states infinitely often will satisfy the Until eventuality condition.

(Refer Slide Time: 35:24)



Here is the final automaton, step 1 we look at all Compatible states we list down all the sub formulae and then this will give rise to many columns which are going to be our states and we remove incompatible states and to this set we add a new state q0 this is giving us the states of the automaton. Step 2 build compatible transitions, transitions that are compatible with the word AND-NOT X Until criteria.

If the LTL formula does not contain any Untils then you can make every state to be an accepting state, because these two conditions already satisfy the properties needed for an accepting expansion, only Until has this extra eventuality criterion. So for each Until sub formula that is present we defined a set of accepting states as shown in the previous slide, if no Until sub formula every state is accepting.

Otherwise there are some Until sub formulae for each Until sub formula define an accepting set, so this will be a Generalized NBA. A run would be accepting if some state in each of these F is visited infinitely often and if that is the case for every Until formula with eventuality criterion would be satisfied. We have seen in unit six module four that every generalized Buchi automaton can be converted into an equivalent Buchi automaton you could refer this module to see the procedure.

In general the algorithm that we have mentioned could give automata which are exponential in the size of the formula, because what we do is that we list down the sub formulae and then we look at all possible values there might be incompatible states but in principle this could lead to an exponential size for the automaton.

Over the years many algorithms have been proposed which performed better than this algorithm, however it has been shown that there are some LTL formulae for which any algorithm will give Buchi automaton with exponentially many states this finishes the explanation of the classic LTL to Buchi automata construction the best way to understand this algorithm is to work it out on many small examples.