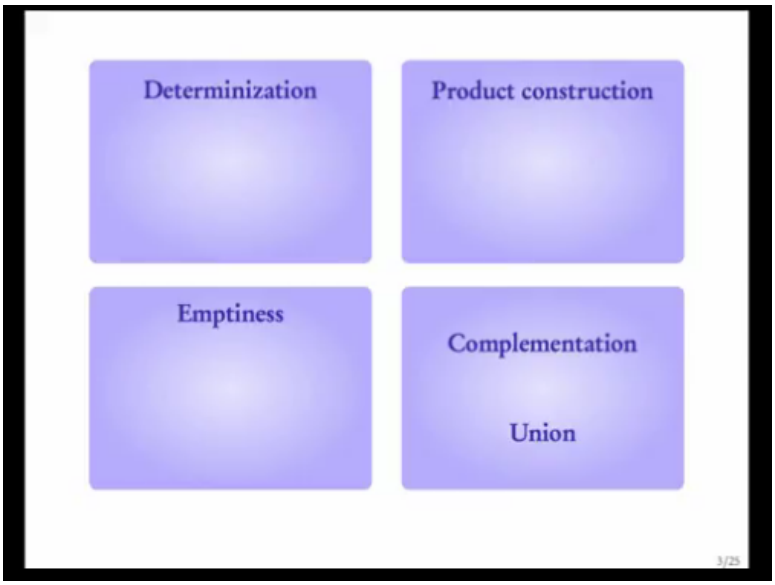
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 20
Simple Properties of the Finite Automata

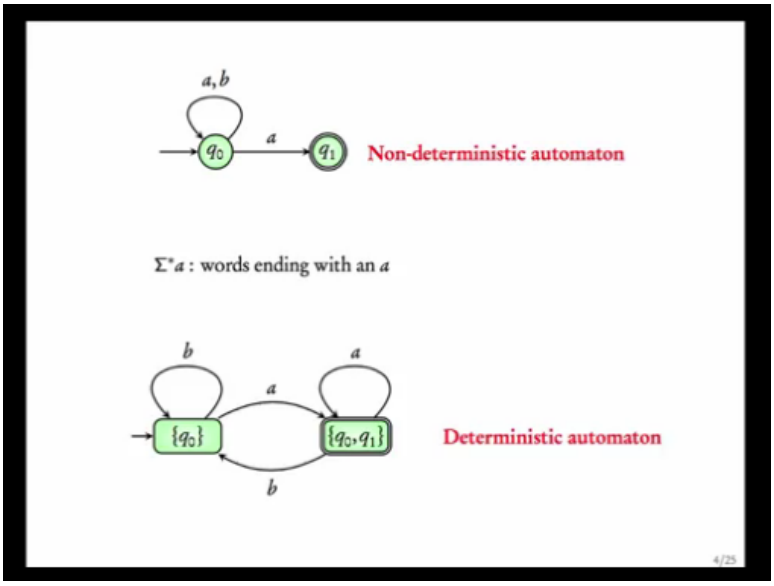
In the last module we gave an introduction to finite automata DFA'S and NFA'S.

(Refer Slide Time: 00:17)



In this module we will be seeing some simple properties of finite automata. There are 4 parts to this module in the first part we will see what is called determinization?

(Refer Slide Time: 00:28)



Look at this NFA non-deterministic finite automaton. Why is this non-deterministic? In this state on an a the automaton can choose to stay q0 or it can go to q1. What is the language of this automaton? We need to look at all paths that end in an accepting state. When can a path end in an accepting state? You start here this is my initial state; you can keep doing this transition as long as you want and then at some point you take this transition which will take to you to an accepting state.

So, all paths from the initial to accepting consist of words which end with an a. You take any word which ends with an a; for example a, b, b, a you will find a path that ends in an accepting state for example a, b, b, a okay. So, I hope you are convince that this NFA accepts the language sigma star a; sigma star a is the set of all words that end with an a. Now, I will explain some construction you will understand what it is finally.

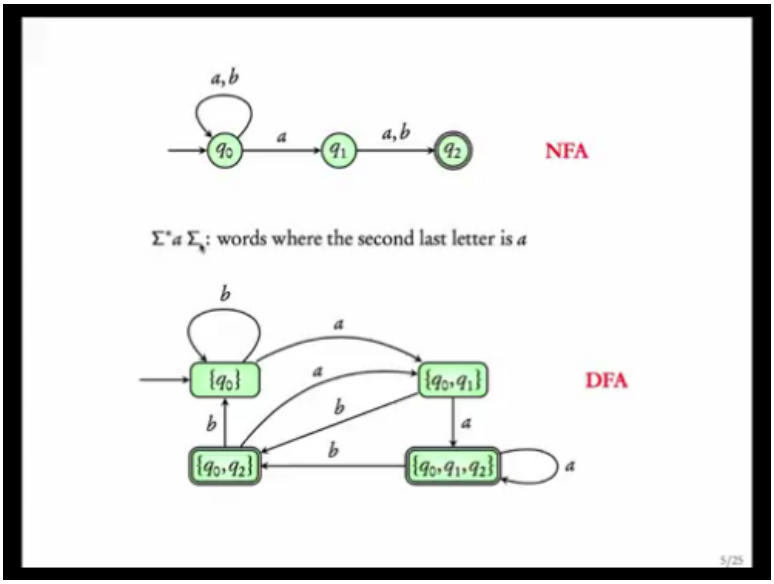
Suppose i consider the state q0; from q0 on an a the automaton can either stay at q0 or go to q1. Let me denote this fact like this from q0 on an a you go to a set q0 comma q1 forget why this is marked like this just for the moment notice that from the set q0 if the automaton sees an a it goes to the set of states q0 comma q1. At q0 when the automaton sees a b what happens it can go back to q0 only. So, from the set q0 if you see a b you have to go back to the singleton set q0 again.

Now, consider this set from q0 on an a what happens you go back to q0 q1 from q1 there is no transition on an a. So, from this set if a is seen the automaton will stay in this set itself. Now, from q0 q1 if the automaton sees a b see from q0 if the automaton sees a b it stays in q0, from q1 there is no b. So, from this set of states if b is seen the automaton goes back to the set q0. This looks like an automaton in itself with states marked by set of states from this automaton.

Now, what is the initial state? Here the initial state is q0 so here we marked the initial state to be the set q0. Look at this automaton; Can you say what is the language of this automaton? Let's look at some accepting paths. Now, this is you see b any number of times the moment you see an a you go here. So this is a word that ends with an a; as long as you keep seeing a you stay here the moment you see a b go back and you stay here till you keep seeing b's and you came back here only when you see an a.

Now, this automaton also accepts the same language of words that end with an a. The extra property of this automaton is that it is deterministic. Let me tell you why we marked this state to be accepting; if you look at this set there is 1 state which is accepting so you should marked this as accepting. Let me give another example and the construction will become clearer.

(Refer Slide Time: 05:42)



Look at this NFA, it has 3 states first of all why is that non-deterministic. At state q0 there are 2 choices on an a either the automaton can stay here or the automaton can go here. Once it goes here it can read only 1 more letter so the set of paths that reach an accepting state have the second last letter to be a. For example look at the word b, b, b, ab, a, b so the second last was a. One more example b, a, a you look at all paths that go to q0 they will be of the forms you keep circling here take this and then take this.

So, the set of words accepted by this NFA is sigma star a sigma I hope this is clear. Now, let us try to find an equivalent deterministic automaton using the construction that I explained in the previous slide. We start with the set q0 which will be our initial state this is the only state in the NFA. From q0 on an a this NFA goes to q0 and q1 we marked like this so we will have another state marked with the set q0 comma q1; q0 on a b can go only to q0 so we have this transition.

Let us look at this state from q0 on an a you can go to q0 and q1; from q1 on an a you can go to q2. So, from the set q0 q1 on an a you can go to this set of states q0, q1, q2 and since q2 is marked as an accepting state this state would be marked accepting. Now, from q0 comma q1 on a b where can we go q0 on a b goes to q0; q1 on a b goes to q2. So, this set will go to q0 comma q2 and since q2 is an accepting state this state would be marked as accepting.

So, you should be looking at the set you should see if there exist at least 1 state which is accepting and in which case it has to be marked accepting. Let us continue drawing the transitions from q0, q1, q2 on an a you have to stay in this set, on a b note that q0 on a b goes to q0, q1 on a b goes to q2 and q2 has no transitions from b. So, overall from q0, q1, q2 on a b you go to this set of states. Now, what about this set? from this set on an a where can we go; q0 goes to q0 q1, q2 cannot go anywhere.

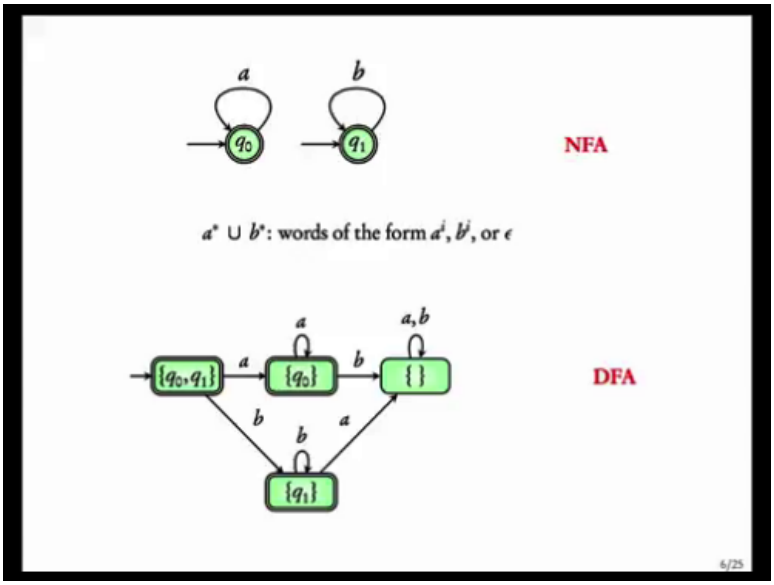
Hence this set on an a goes to the set q0 comma q1. Now, what about b from q0 on a b you stay only in q0 and from q2 you cannot take a transition on a b. Hence from q0 q2 on a b you go here this is now deterministic automaton from this state on an a there is only 1

choice, on a b there is only 1 choice same for every state there is a single initial state so this automaton is a DFA. Let us try to understand the language of this DFA; the language is the set of the paths that reach an accepting state.

Now, if you see there are 2 accepting states; from this accepting state I mean how do you reach this accepting state. The only way to reach this accepting state is through this and this can be reached via this. And once you are here you can keep doing this transitions no matter whatever you do either you come from here or you keep circling the second last would be a.

So, all words that end up here will have the second last to be a. Let's see examples b, b, b, a, a. And if you see more a' s still the second last is going to be a. Now, what about this b, b, b, a, b or a, a, b or once you are here b, a, b. So each time you come here the second last would be a. Let's see 1 more example b, a, b, b, a, b so you can have a look at this automaton and convince yourself that all paths from the initial to an accepting state would have the second last letter to be a hence the language of this automaton is going to be sigma star a sigma.

(Refer Slide Time: 12:14)



Let us look at another example; this entire thing is 1 NFA with multiple initial states there are 2 initial states here q0 and q1. The automaton can non deterministically choose to

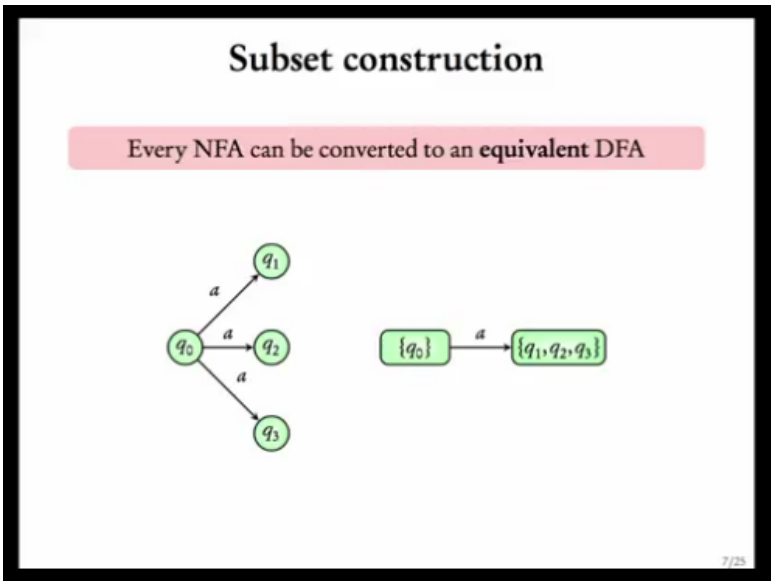
start from either q0 or from q1; if it starts from q0 it can see only a's, if it starts from q1 it can seen only b's. What is the language of this automaton? Any guesses, it has words of the form a star b star and of course epsilon because the initial state is accepting. Let us now try to find an equivalent deterministic automaton.

Now, the initial state here would be the set q0 comma q1 since both of them are initial and since q0 is final this is marked as accepting q1 is final as well but it is enough that one of these is accepting here for this set to be marked accepting. Okay, now what about the transitions from this set lets start to take an a q0 on an a stays at q0, q1 on an a cannot do anything q1 cannot see an a. Hence the set of states that we see on an a is just q0; since q0 is accepting this state is marked accepting.

Now, what about b from here if you see a b you can go to this set q1; from q0 on an a you stay here, from q0 if you see an a you keep staying in this set. Now, what happens if you see a b from q0; you cannot see a b from q0 on a b there is no transition. How do we represented here you denote a state marked by the empty set q0 on a b goes to this empty set. Similarly, what about this q1 on a b it stays here on an a it goes to this empty set.

Now, this empty set on an a and b will just be empty this automaton will be a DFA and look at the language. If a paths ends in q0 then it has seen only a's, if a path ends in q1 it has seen only b's and since this is an accepting state epsilon is also accepted. When you when you are seeing a's the moment you see b you do not accept when you are seeing b's the moment you see an a you don't accept. So, this automaton will give the same language but it is deterministic.

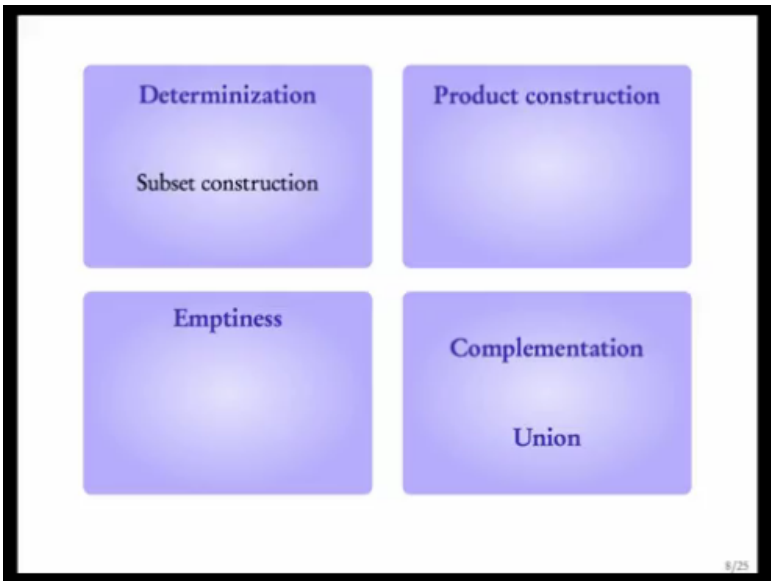
(Refer Slide Time: 15:54)



The construction that I have been explaining over the last 3 slides is called the subset construction. Every NFA can be converted to an equivalent DFA and how you can do it you take any NFA and use this kind of a subset construction. The states in the DFA would be sets of states of the NFA and how are the transitions you look at the set of states here; here there is only 1 state in this state on an a you look at all the states that it goes in the NFA and in the DFA it would be from q0 it would the set q1, q2, q3 this is going to be single state.

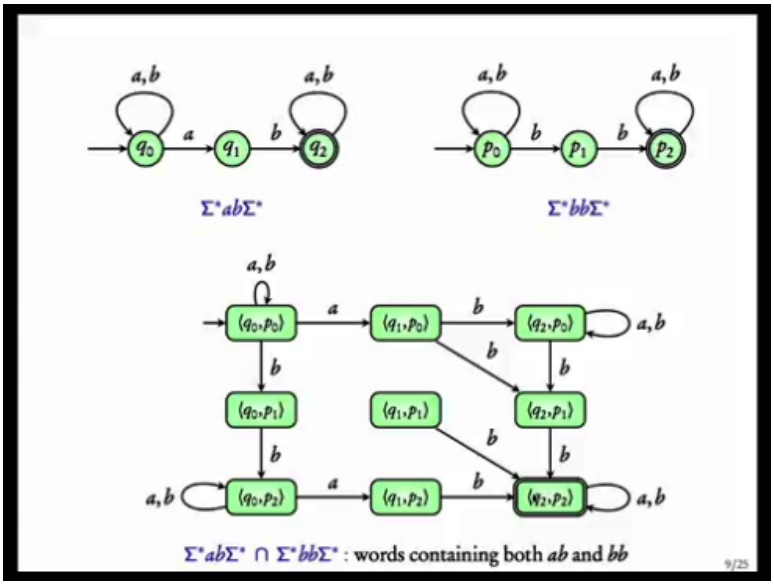
And how do you mark accepting states? Like this in this previous example; as soon as you intersect I mean as soon as the set intersects with an accepting state this has to be marked accepted. And what is the initial state? The initial state is the set here it is more clear; the initial state is the set of initial states of the NFA.

(Refer Slide Time: 17:25)



I hope the subset construction is clear this will tell you that every NFA can be determinized to an equivalent DFA. The next construction that we will see is called the product construction.

(Refer Slide Time: 17:41)



Consider these 2 NFA'S; this is an NFA because there is a non-deterministic choice at q0 at q1 you do not know what to do on a. Now, what is the language of this NFA; it accepts all words that have a b inside. If you look at the accepting parts as an all parts that go from the initial to the accepting state they should have crossed a b. Similarly, this automaton accepts all words that contain b b so the language is sigma star b b sigma star; here it is sigma star a b sigma star.

Now I will explain a construction similar to the interleaving that we saw before we will see what is the language that it gives. Firstly, the states are going to be a cross product in the sense; for every state here you look at all states here and then follow product in the sense q0 p0, q0 p1, q0 p2 then q1 p0, q1 p1, q1 p2, and then q2 p0, q2 p1, and q2 p2. The initial state is given by the state where both of these are initial q0 is initial, p0 is initial so q0 p0 would be initial.

Now, carefully listen the accepting state is given by the state where both of them are accepting q2 p2 it's not the case that if one of them is accepting you mark it accepting no. You find the state where both of these are accepting and then you make it accepting. Firstly, see that the states are different from the subset construction you are not forming subsets; firstly there are 2 NFA's and you are just taking the product of the states. In the sense every state consists of a state of NFA1 and state of NFA2 that's all similar to the interleaving.

Now, what are the transitions? On q0 on an a NFA1 can either go to q0 or to q1, from p0 on an a NFA1 goes to only p0. So from this state there is a choice from q0 p0 on an a either you choose to stay at q0 here i mean i have done in for both a and b because q0 on an ab can choose to stay at q0 and p0. Now, on an a it can also do this q0 on an a can go to q1, p0 on an a can stay back. On a b q0 can do what? It can either stay here well there is only choice on a b; q0 on a b stays here, p0 on a b can choose to stay here or go here.

If it chooses it to stay here I mean if it chooses it to stay at p0 that is limit by this transition if it goes to p1 it is limit by this transition. Firstly, we are not constructing a deterministic automaton we are doing something else the states are clear to you now for the transitions what we do is you see from q0 on an a what are the possibilities; q0 on an a can go to either q0 or q1, p0 on an a can go only to p0.

So q0 p0 on an a can go to either q0 p0 or q1 p1 sorry q1 p0. What about b? q0 on a b can go only to q0, p0 on a b can go to both p0 and p1. So q0 p0 on b can either go to q0 p0 or

q0 p1. Let us continue doing this look at this state q0 p1; On a b what happens? q0 of course can go only to q0, p1 can go to p2. So q0 and p1 simultaneously can go to q0 p2.

So, if you see the transitions are taken by both of them from q0 p1 on a b q0 stays in q0, p1 goes to p2. Let us look at this state what happens now this is limiting the fact that NFA 1 is in q1, NFA2 is in p0. What happens when they seen an a? q1 cannot move so there will be no transition on an a from this set.

If they are here q1 p0 and both of them have to move it can be only on a b and on a b what happens q1 goes to q2, p0 can go to either p0 or p1. So from here you have this transition where q1 goes to q2 and p0 stays in p0 and the other transition where q1 goes to q2 and p0 takes this transition to go to p1.

Note that both of them are taking the transition here q1 is taking the transition to q2 and p0 is using this transition to stay in p0. Now, let us look at this state we want both of them to simultaneously take the next transition now q0 p2 this is this means that NFA1 is in q0, NFA2 is in p2. What are the next transitions? Of course on an a or b you have this transition, you have this transition.

More over on an a q0 can go to q1; hence you have these 2 transitions here when you are here so essentially this product construction is limiting the movement of both the NFA's. When you read a word what happens to both of them that is what this is limiting so from q0 p2 on an a either both of them can stay in the same state. Similarly, on a b both of them can take the transition to stay in the same state or on an a q0 can go to q1 and p2 takes this transition to stay back here this will give us q1 p2.

Now, look at this from q1 comma p2 you can't read an a why? because q1 cannot read an a. The only thing both of them can read is b and q1 on a b goes to q2, p2 on a b stays in p2. Now, look at the state q2 p0 what happens on a b; p0 on a b can go to p1, q2 can stay back here so we have this and of course we will have this as well because both of them can take the transitions to stay in the same state. And one more from p1 you can read only

a b so from q2 to p1 you can read only a b we are not done yet look at the state we have to give transitions for this state.

Now, q1 on an a cannot do anything so from q1 you cannot read an a so from this state you cannot read an a on a b what happens q1 goes to q2, p1 goes to p2. Now, from this state whatever you read you stay here because q2 stays here, p2 stays here. Now this automaton is called the synchronized product of these 2 NFA's this need not have to be deterministic. What this automaton does is that it starts from both the NFA's and simultaneously reads a word.

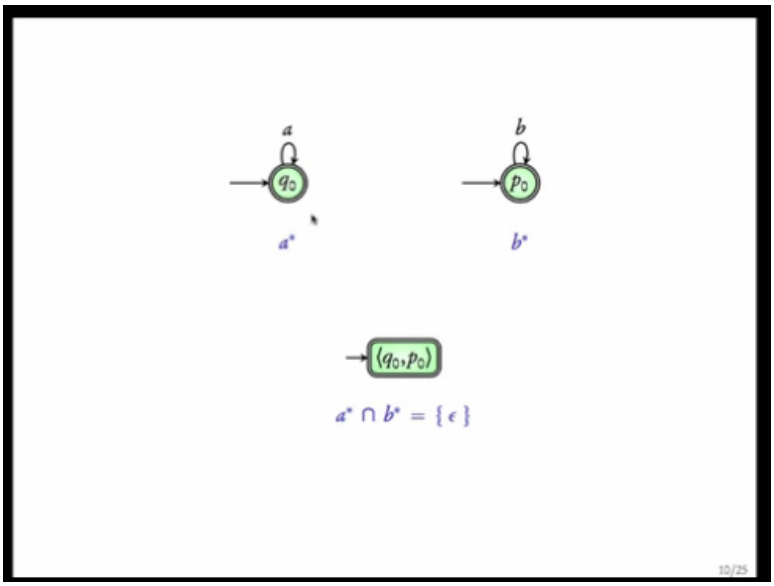
For example if you read a word ab, ba lets see ab, ba this is one possible way of reading the word on this automata ab, ba. Now, what about the same word here ab, ba so this is p0, p0, p1, p2, p2. This automaton it starts from a q0 p0 what we did was we did we did this path, So, on an a you go here q0 goes to q1, p0 stays in p0; on a b q1 goes to q2 and p0 goes to p1 so you come here and then what.

We did this one and we did this one so you go to q2 p2 and then you take this transition. It is giving us ways in which both the automaton simultaneously move forward and if there is a state where both of them are accepting it means that both the automata have a path from the initial state to reach an accepting state. For example if there is q2 p2 this means NFA1 has a way to go from q0 to q2 and NFA2 has a way of reaching p2. Since both of them are accepting the path that takes us here will be a word which is in the intersection of these 2 languages.

Hence this automaton would represent the set of words that contain a b as well as b b; for example a b, b b it contains both a b and b b. Of course a, b, b, b this contains a b and it also contains b b. Let us look at other parts b b a b so you do whatever you want here and then when you reach this state you would have definitely read a bb as well as an a b. No matter how you reach it you would have read both b b and a b of course you have to read starting from the initial state not from here this is an unreachable state from the initial state.

So, from the initial state you look at all parts that go to q2 p2 you will see that will contain both a b and b b.

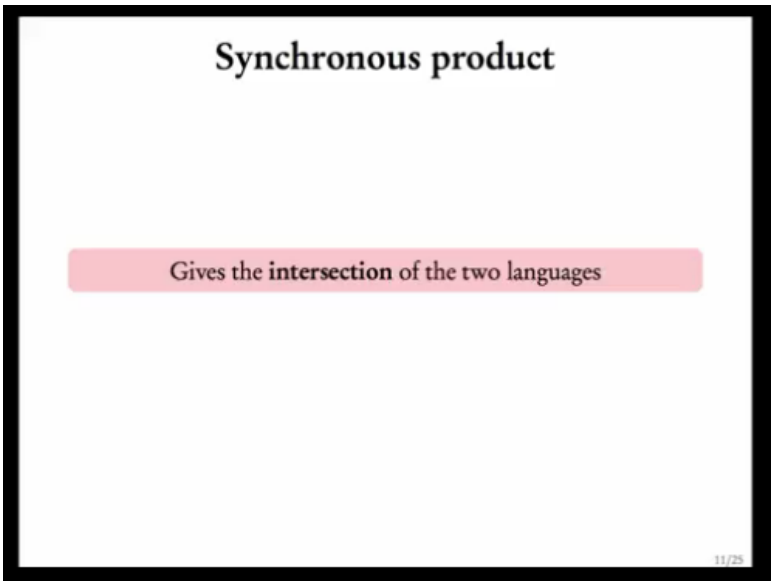
(Refer Slide Time: 31:08)



Let us see another example this is NFA1 this accepts a star this is NFA2 this accepts b star. Let us now take the product there is only 1 state which will be q0 p0 both of them are here now what happens on an a q0 can read an a but p0 cannot read an a so you cannot have a transition on an a from this joint state. Similarly, q0 cannot read a b but p0 can read a b however on this joint state you cannot read b together.

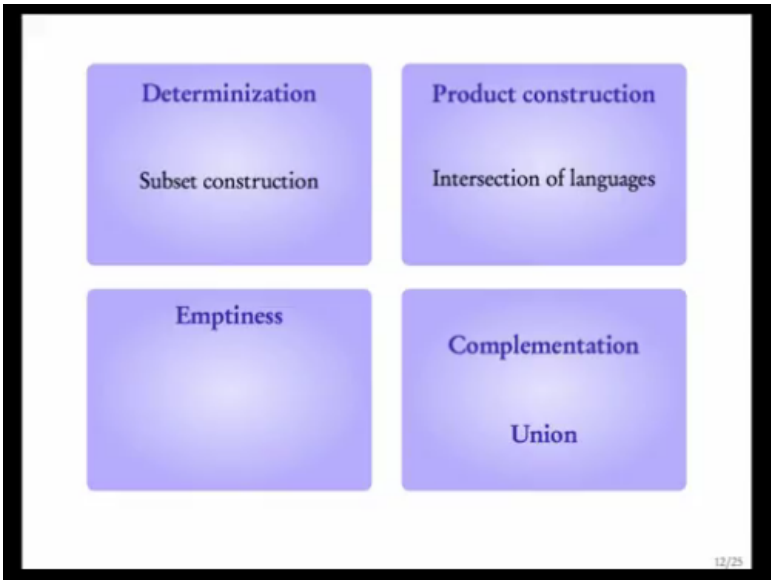
So, this is going to be the synchronized product of these 2 automata and what is the language of this is exactly epsilon which is the intersection of a star and b star.

(Refer Slide Time: 32:17)



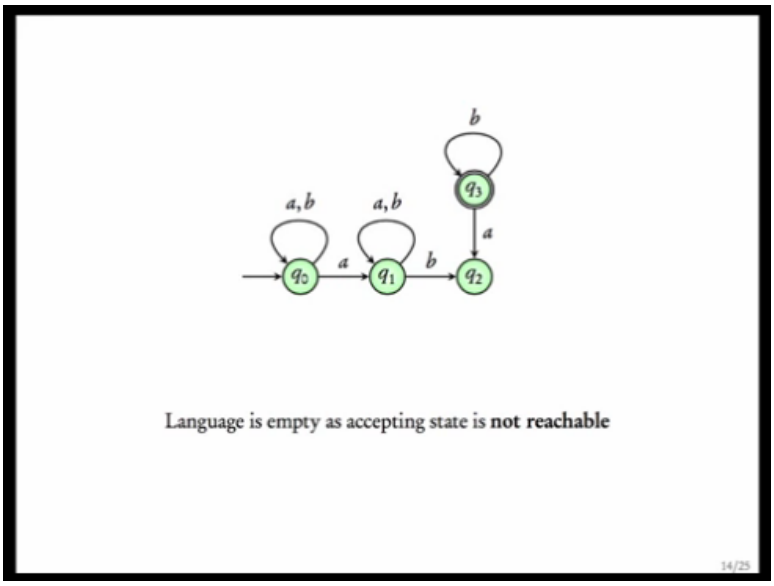
The construction that i explained now is called synchronous product. I have explained the construction through an example i did not want to give the formal notation. However, i hope that construction is clear based on the example itself.

(Refer Slide Time: 32:39)



We have seen 2 constructions the first was called the subset construction given an NFA we use this construction to get an equivalent DFA. The other construction was called the product construction given 2 NFA's you use this product construction to get the NFA describing the intersection of the 2 languages given by the NFA's. The next thing that we are going to see is called emptiness problem let see what it is.

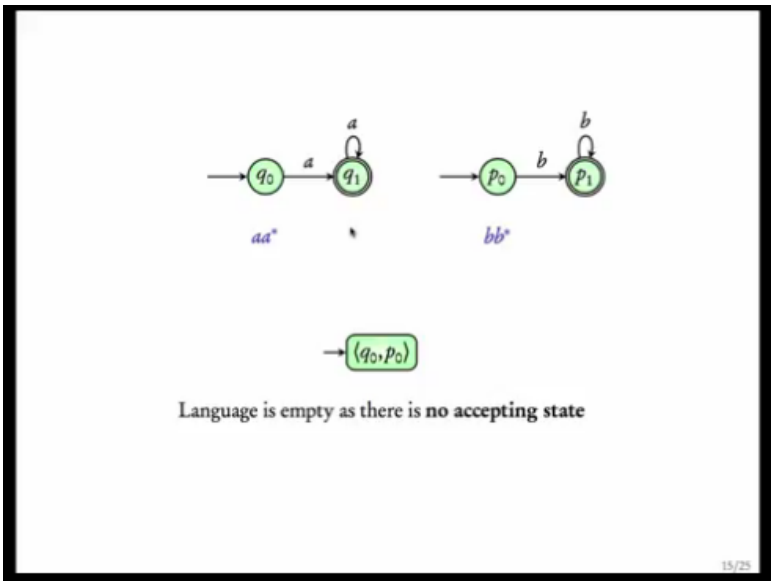
(Refer Slide Time: 33:23)



Look at this NFA does not have an accepting state? No. So what does it mean it just means that the language is empty the definition of the language is the set of all parts that go from the initial state to an accepting state. But when there is no accepting state at all there is no path to an accepting state and hence the language of the automaton is going to be empty.

Look at this automaton it does have an accepting state however there is no path to this accepting state from the initial state there is no way you can go from q0 to q3. Hence there is no path from q0 to q3 implying that the language is empty; language is empty as accepting state is not reachable.

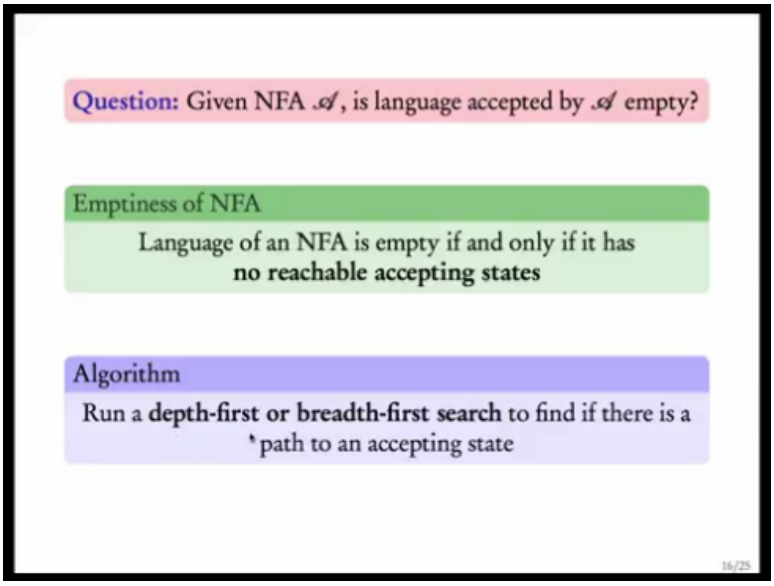
(Refer Slide Time: 34:35)



Now, look at these 2 NFA's this accepts a a star, this accepts b b star. Let us now take the synchronous product you start from q0 p0, q0 can read an a but p0 cannot read an a. So, you cannot have a transition on a here p0 can read a b but q0 cannot read a b. So you cannot have a transition from b out of this state hence the synchronous product is just this much. Since q0 is not accepting this cannot be made an accepting state symmetric argument p0 is not accepting as well so this cannot be made accepting.

Hence, the language is empty for this automaton which is in fact the synchronous product this is clear because this language is a a star, this language is b b star and a a star intersection b b star is empty. This is a set of all words of the form a power n where n is bigger than or equal to 1 this is of the form b power n where n is bigger than or equal to1 there can be no word in the intersection of these 2.

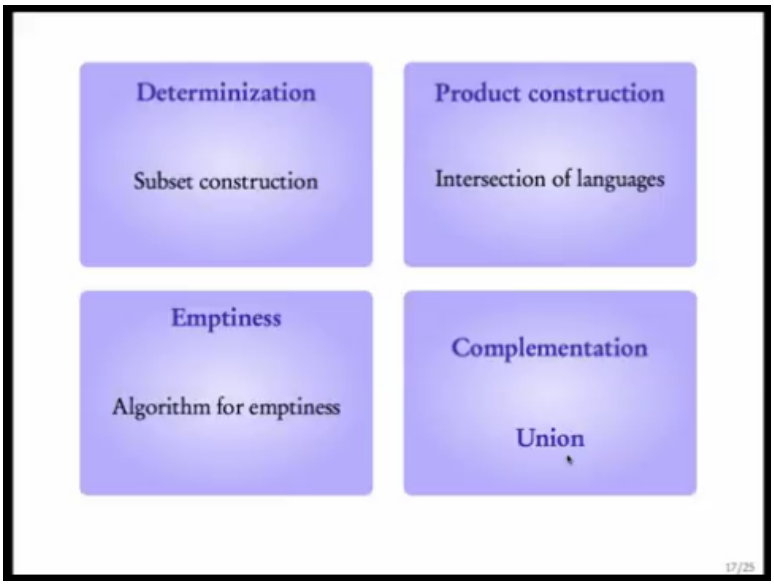
(Refer Slide Time: 36:13)



Let us now asked this question suppose i give you an NFA a; when would you say that the language accepted by this NFA is empty. From the examples you would understood that the language of an NFA is empty if and only if it has no reachable accepting states given the NFA you need to just check if there is an accepting state which is reachable. If so the language is not empty but all the reachable states are not accepting then the language is empty.

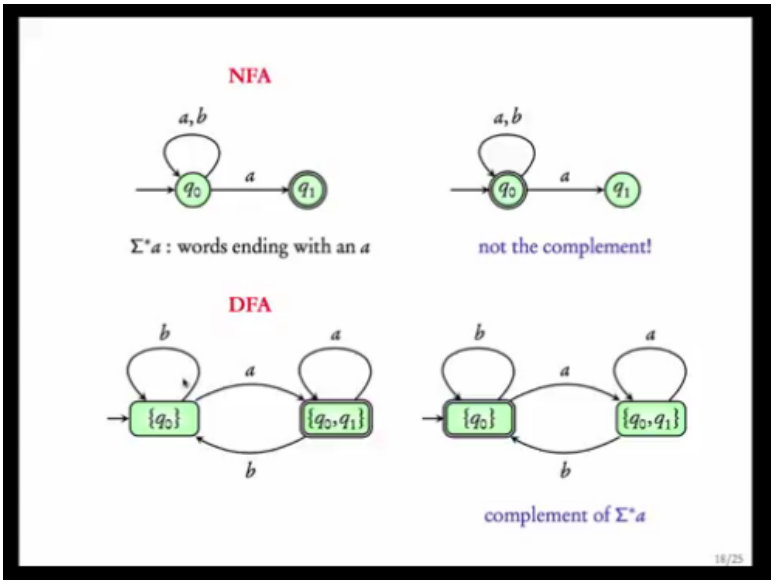
How would you check this? You can use either a depth first or breadth first search to check if there is a path to an accepting state. For the depth first you could have a look at Unit 03 Module 03 on invariants we would have explained a depth first search to check invariant properties here instead of checking invariant properties you are just checking if there is a path so a standard depth first or breadth first would do.

(Refer Slide Time: 37:40)



The point to note is that given an NFA there exists and algorithm for checking if the language of that NFA is empty this would be very useful in the later units. The final property of automata that we would be looking at in this module involves complementation and union.

(Refer Slide Time: 38:08)



Look at this NFA which we saw before; this is the NFA which accepts all words that end with an a and this is the corresponding DFA. Let me do a modification to this DFA; let me interchange the accepting and non-accepting. This state was not accepting so I will make it accepting, this state was accepting so I will make it non-accepting.

Try to look at the language of this DFA all paths that starts from q0 and end in q0 will definitely end with the b. Look at this; this bbb, a you see a any number of times and then when you get back to q0 you have to read b. So the language of this DFA obtained by interchanging the accepting and non-accepting states is exactly the complement of this sigma star a.

So this is going to be all words which are not in sigma star a they would be the set of all words which end with the b or just epsilon. So, epsilon is accepted because q0 is accepting and all words that end with the b or also accepted. Let us now try to do the same trick with this NFA; let me interchange the accepting and non-accepting states.

What we would get q0 will become accepting and q1 will become non-accepting is the language of this NFA the complement of this. That is does this accept the set of words that end with a b only; it's not true, because you see this NFA can accept any word. So, the language of this NFA is in fact sigma star and it is not just sigma star b as the language of this automaton is.

So, the trick of exchanging the accepting and non-accepting states on a DFA would give you the complement language. However, that trick does not give you the complement in an NFA here you don't use the trick in NFA at all. Let us try to understand why it doesn't work, look at the word a b a it belongs to this language.

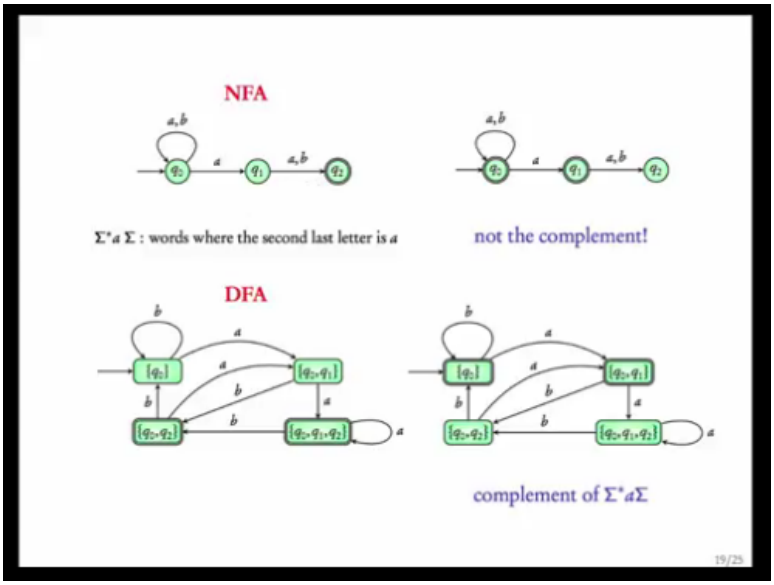
Let us look at the paths from let us look at all possible ways that the NFA can read a b a it can do this a b a or it can do a b a. Since in the other case the a b a it went to an accepting state the word is accepted by the NFA. However, it does not mean there are other paths that end on an non-accepting state on the same word. So, this word a b a can either go here or stay here.

So, when you are interchange you will always you will also be accepting words that are accepted by this and hence it's not going to be the complement of this language. On the

other hand if you look at the DFA; let us look at what happens on a b a, a b a there is a single path on a b a.

This is the property of a deterministic finite automaton on every word the automaton has a unique run it is called every word there is a unique way of reading it and hence there is a unique end to that word. So, if you interchange you will be reading only the words rather you will be accepting only the words which were not accepted by the DFA.

(Refer Slide Time: 43:00)



Let us look at another example; this was the NFA which accepted sigma star a sigma all paths to the accepting state are of the form sigma star a sigma. This is the equivalent DFA; when you interchange the accepting and non-accepting states so q0 this set become accepting, this set become accepting. Let us see what kind of words that reads; it reads words a a b a or a b a b a. If you see this automaton would accept words in the complement of this what are the words which are not in this language.

They are the words where the second last letter is b additionally epsilon the singleton a and the singleton b is also accepted. So, for example epsilon is accepted the word a is accepted, the word b is accepted and all the words where the second last is b will also be accepted for example a b a the second last was b. Now a a b b a so the second last was b now you do any amount of b b b a b a b a again the second last was b.

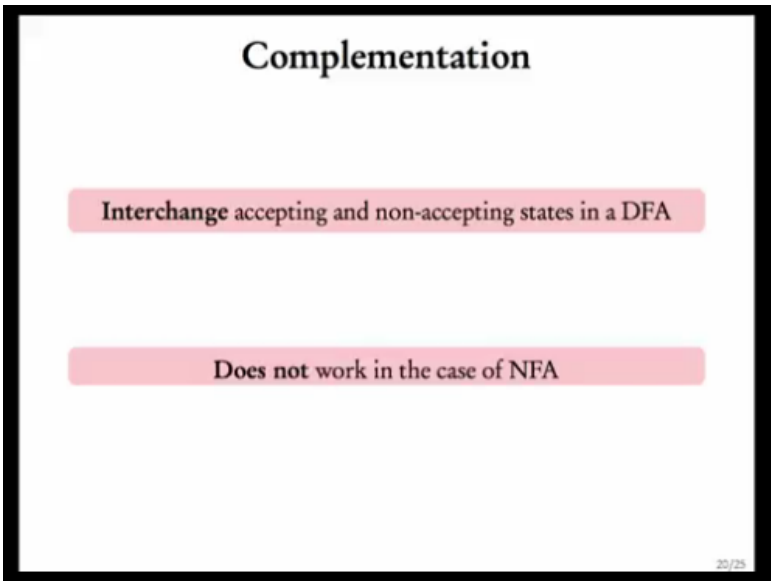
The property of the DFA is that for every word there is a unique run. So all runs that end with an accepting state would be accepting and all runs which end in an non-accepting state would not belong to the language. So far every word there is a unique run and if that word belongs to that language it will end in an accepting state and if the word does not belong to the language it will end in a non-accepting state.

Hence when you interchange the accepting and non-accepting states you will essentially be accepting the complement of the language I mean complement of this language. However, the same trick doesn't work with an NFA for example c if I interchange the accepting and non-accepting states what do i get not the complement why because i can still accept words in this language.

For example a a a is here; so here a a a is accepted because I can do a a a, however I could have also done a a a. So this automaton which is which is obtained by interchanging the accepting and non-accepting states need not accept the complement it would in general these something else. The point to note is that for every word an NFA can have multiple runs there could be runs which end in an accepting state, that could be runs which end in non-accepting state.

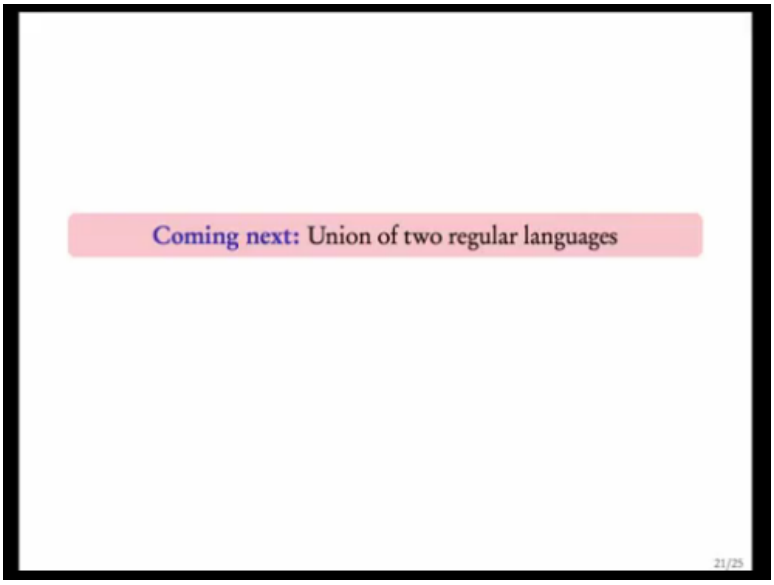
As long as there is 1 run which ends in an accepting state we will accept that word please go through this argument again and again.

(Refer Slide Time: 47:01)



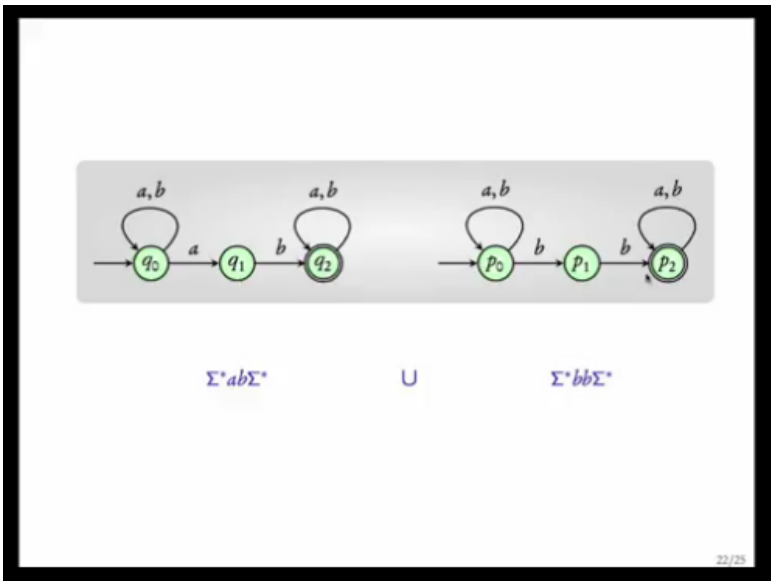
So what we have seen that when you interchange accepting and non-accepting states in a DFA you get the complement language and the same time that does not work in the case of NFA.

(Refer Slide Time: 47:17)



What we will see next is given 2 automata how do you construct the automaton representing the union of these 2 languages.

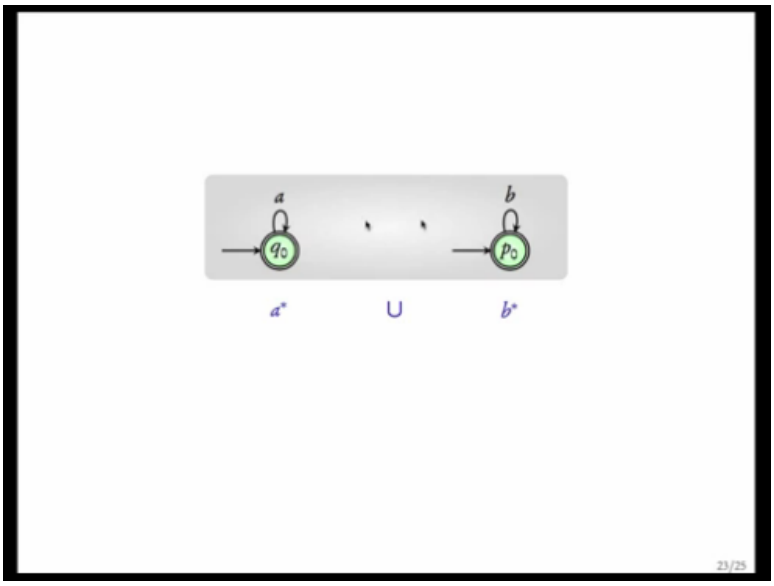
(Refer Slide Time: 47:28)



Consider these 2 NFAs which we which we have seen before this is the NFA for sigma star ab sigma star this is the NFA for sigma star b b sigma star this is language 1, this is language 2. Now, what is union of 2 languages; we look at words here, we look at words here combine everything you get a bigger set. So, the union of these 2 would be the set of all words containing either an ab or a bb.

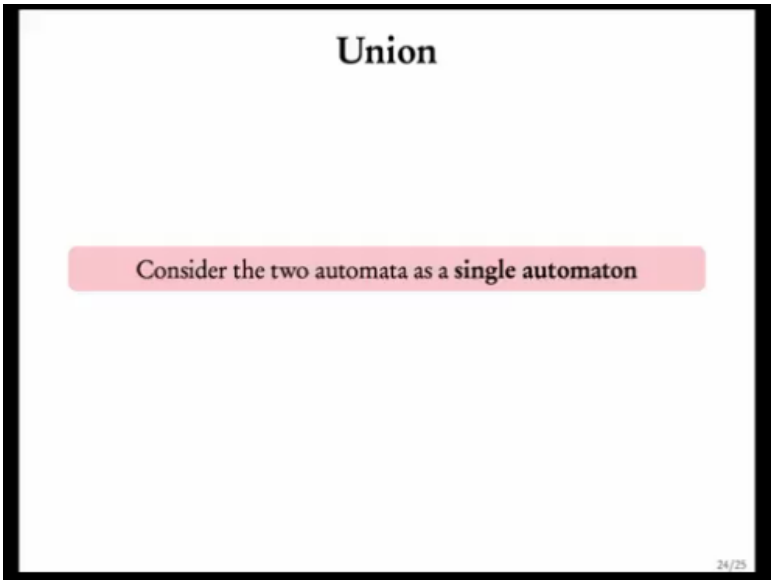
What would the automaton for the union b it is very simple you look at these 2 as 1 automaton. This is called the disjoint union of the automata this full automaton contains 6 states there are 2 initial states q0 and p0. So this would be an NFA and this automaton will accept the union of these 2 languages. On a word you can choose to start from here or you can choose to start from here. So, all accepting paths from here belong to the language and all accepting paths here also belong to the language hence you get the union.

(Refer Slide Time: 48:45)



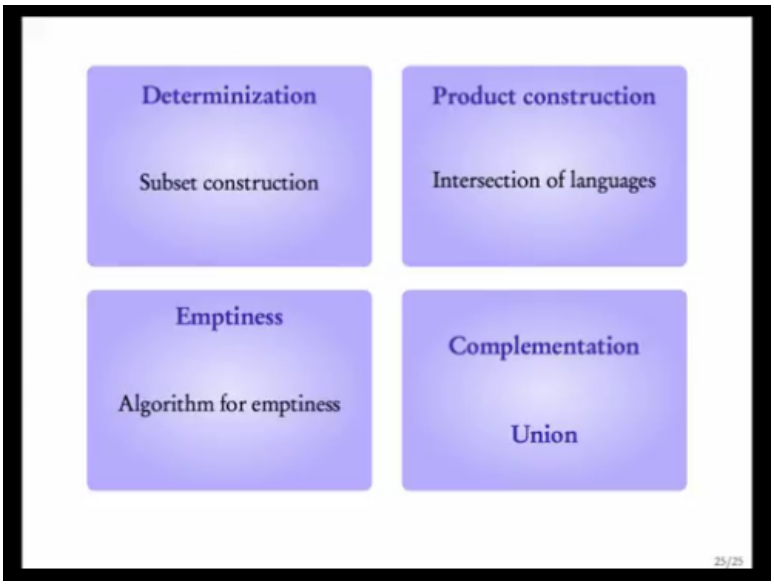
Similarly, this is a star this is b star and this was the automaton for a star union b star this is called disjoint union. This automaton has 2 states and both of them are initial.

(Refer Slide Time: 49:02)



So, union is very simple given to NFAs consider the 2 automata as a single automaton this called a disjoint union.

(Refer Slide Time: 49:14)



This brings us to the end of this module in this module we have seen 4 important properties of automata through lot of examples. The first property was the first property was the fact that every NFA can be converted to an equivalent DFA. We saw what it is called the subset construction. Secondly we saw a construction called synchronous product and this synchronous product gave us the automaton representing the intersection of the 2 individual languages.

Thirdly we saw the given an NFA there exists an algorithm to check if the languages is empty. And finally, we saw how to construct automata for the complementation and union; for the complementation you should take a DFA and then interchange the accepting and non-accepting states, for the union you just take both the automata and consider the whole thing as a single automaton. We will of course get an NFA because they will be multiple initial states.