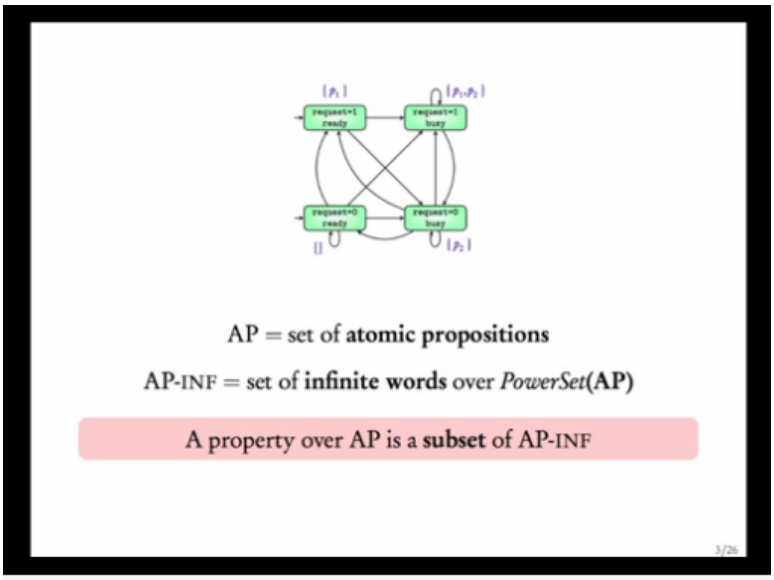
Model Checking
Prof. B. Srivathsan
Department of Computer Science and Engineering
Indian Institute of Technology – Madras

Lecture - 19
A gentle introduction to automata

Welcome to module 2 of this unit, as we saw in the previous module the goal in the rest of this course is to understand the algorithms used inside the model checking tools. In order to understand these algorithms we need to have a good understanding of this concept called automata. Some of you might have already seen the concept of automata in your courses. However, this video assumes that you have no idea about automata and I will gently introduce this concept using examples.

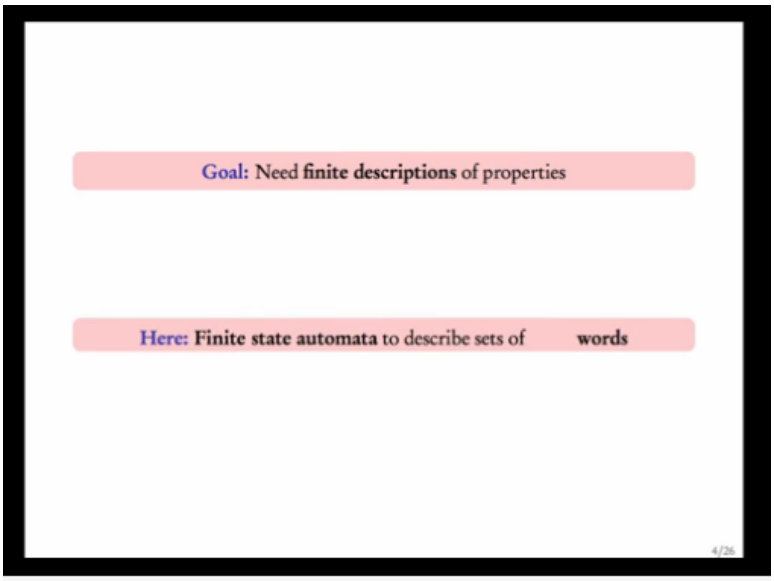
(Refer Slide Time: 00:54)



What are properties? A property is nothing but a set of infinite words we had seen this in module 2 of unit 3. However, the number of words in the subset could be infinitive, for example if the property says that P1 is true always then you can have infinitively many words. Why there is 1 word where every letter consists of just P1 there is a word where the first letter contains P1 and P2 and the rest of them contain only in P1.

In another word the first 2 letters contains P1 and P2 and the rest of them contain only P1and so on. So, there are potentially infinitely many words that can satisfy a property.

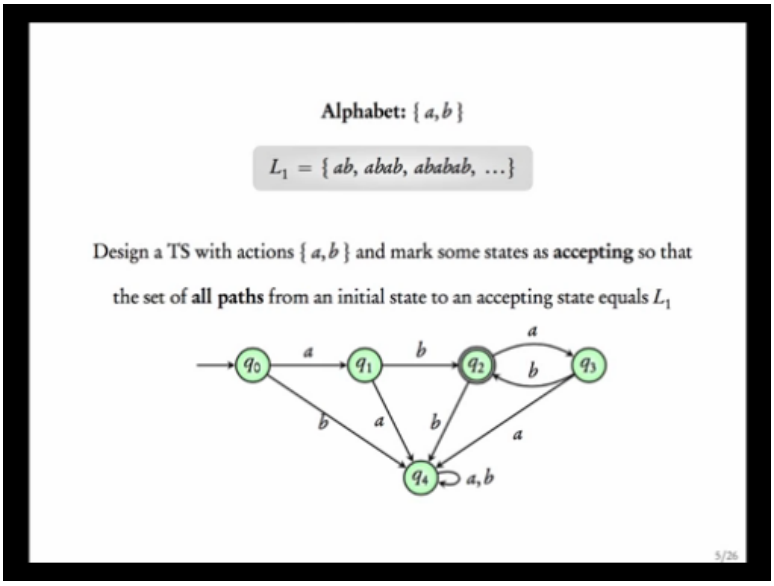
(Refer Slide Time: 02:02)



We need finite descriptions of these infinite sets, some of the finite descriptions are GF etc. Here we will be using what are called finite state automata to describe sets of words. In particular here we will concentrate on finite words, in the next unit we will talk about infinite words. Let me repeat the goal of this module is to get finite descriptions of set of words. The sets of words could be infinite in a set there could be infinitely many words.

However, we want to characterize them using some finite names and here we will talk about finite state automata. We will see a lot of examples and you will understand what I mean by a finite description of a set of words.

(Refer Slide Time: 03:16)



We are talking about words, to form words you need an alphabet for example in English there are 26 letters in the alphabets and words are formed out of these 26 letters. Similarly, let me take some alphabets which is restricted to the letters a and b there are only 2 letters in the alphabet and now I am forming a set of words using this alphabet.

So the words are ab, abab, ababab and so on. There are other words like bab, bbb etc but I am not considering those words I am just considering a particular set of words and I am naming it L1 and the words here are of the form ab concatenated some n times where n is from 1 to infinite ab, abab, ababab and so on. Okay, so I have a set of words now what do I want to do with this. I want to design a transition system, you know what is a transition system is with actions a, b.

So, the transition system should have state transition and actions should be only a and b and mark some states of the transition system as accepting. So, I want to design a transition system with actions a and b and secondly I want to mark some of the states of this transition system as accepting so that the set of all paths from an initial state to an accepting state equals L1.

What do I mean by that once I designed this transition system and mark some state as accepting I will look at all parts that can take me from some initial state to some accepting state. It will some ab, abab depending on whatever the transitions are and let me look at all the parts. My

requirement is that in these parts let us design it and you will understand more clearly what I mean by this.

Lets start so I am going to construct a transition system with 4 states q0 is not going to the initial state and q2 is going to be the accepting state. So, to denote an accepting state I have a double circle I start at q0. The goal is that all parts that take me from q0 to q2 should be of this form, should be a word in L1 and for every word in L1 I should have a path from q0 to q2.

Let us start from q0, at q0 you can either see an a or you can see a b. If you see a b that means any path from q4 to q2 will give you a word that starts with b and since in L1 no words starts with b this will be a bad word for you. So, in the rest of the construction we should ensure that q4 has no path going to q2.

Okay, now we are here now if I see a b from q1 I am getting a word a b this is a path from initial to accepting state and what is the word in that path a b fortunately, a b is in L1 so I am safe. However, at q1 if I see an a I should once again go to q4 because no word in L1 starts with aa. Now, I should still keep ensuring that there is no path from q4 to q2. Let us continue, at q2 if you see an a there is still a possibility of coming back to q2 with a b and staying in L1.

However, at q2 if you see a b you are gone because here you have a word a b b but no word in L1 starts with abb. Let us now complete at q3 if you see a b you can go back to q2 and for same reason if you see an a you should go to q4. Let me now explain the paths from q0 to q2 since we have a cycle there are infinitely many paths.

The first path is a b which is in L1, the next is a b a b which is again in L1 the other one is ab ab ab which is again in L1. Each time you take the cycle you are getting another word in L1. All paths from q0 to q2 will give us a word in L1 and similarly for every word in L1 there is a path in this transition system with that word for example if I take the word ababab say 15 times. The path corresponding to this word is a b and take this cycle 14 times.

Once you go to q4 there is no way you can come back to q2 you can put a loop here with any letter a or b. If you end up in q4 it will be a word which is not in L1. I hope this idea is getting clear so we have got what we wanted for L1 let me give another example.

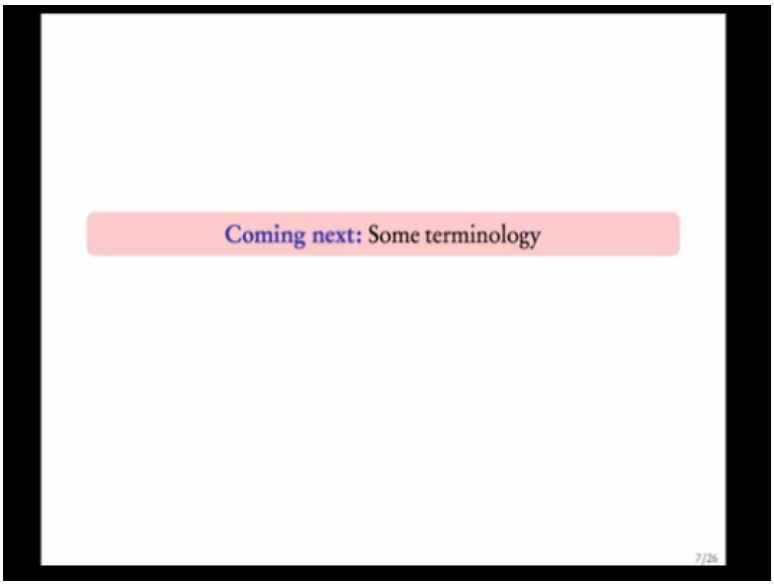
I am looking at a set of words which start with a there are clearly infinitely many words just the letter a this is the word of length 1, aa is word of length 2 that starts with a, ab is again a word of length 2 starts with a aaa aab these are all words of length 3 that starts with a for any length you can give a word starts with a.

So, there are clearly infinitely many words now I want a finite description like this in the similar way. We want to design a transition system so that all paths to the accepting state or words that starts with an a and for every word that starts with an a there should be a path that takes me to an accepting state. Here is the required transition system the initial state is q0 if I read an a I go to an accepting state and in this accepting state I can see any letter from now on because I have read an a in the beginning.

However, if I read a b I should go to a state which is non accepting and then from here whatever I read I should keep staying in this non accepting state. If you look at all paths of this transition system that ending in the accepting state they will be the words which starts with an a and for every words that starts with an a there is a corresponding paths here. This kind of a structure is called a Finite Automaton.

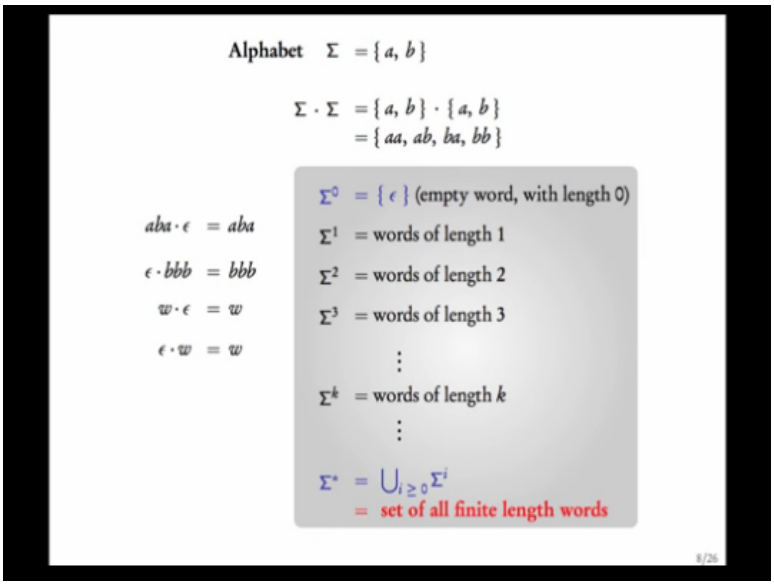
There is not much difference from a transition system the only difference is that we mark some states to be accepting and we are mainly interested in the set of paths that take us from an initial state to an accepting state. This set is what we are interested in for this automaton the set of words that it accepts is L2. Similarly, for this automaton the set of words that it accepts is L1.

(Refer Slide Time: 13:58)



Okay, before looking at more examples of automaton let us put in some terminology which will help us to write things in a succinct way.

(Refer Slide Time: 14:13)



Here is an alphabet usually we get the notation sigma to an alphabet however there is no necessity that you should give it sigma you can call it b q whatever you want. In this module I will call an alphabet with the notation sigma. Now, what is sigma dot sigma I have this set and I want to take dot with this set, what is a dot?

Essentially, I want to concatenate the letters here with the letters here, its like a product what will I get a a a dot b will be just ab, b dot a is just ba, b dot b is bb this dot is just a concatenation operator. When you do sigma dot sigma you will get this sigma dot sigma is essentially giving us words of length 2. We write sigma dot sigma as sigma power 2 so sigma power 1 is just sigma and it's just words of length 1.

So far so good if you extend it sigma dot sigma dot sigma will be sigma cube and this will give us words of length 3. You just take a dot of this set with ab you will get aaa, aab, aba, abb, baa, bad, bba, bbd so this will give us all words of length 3. Similarly, if you do sigma power k that will denote the words of length k. You should just look at it as a notation sigma 1 is just is denoting the words of length 1, sigma 2 is denoting the words of length 2, sigma 3 is denoting words of length 3 and so on for any k sigma k will be words of this finite length k.

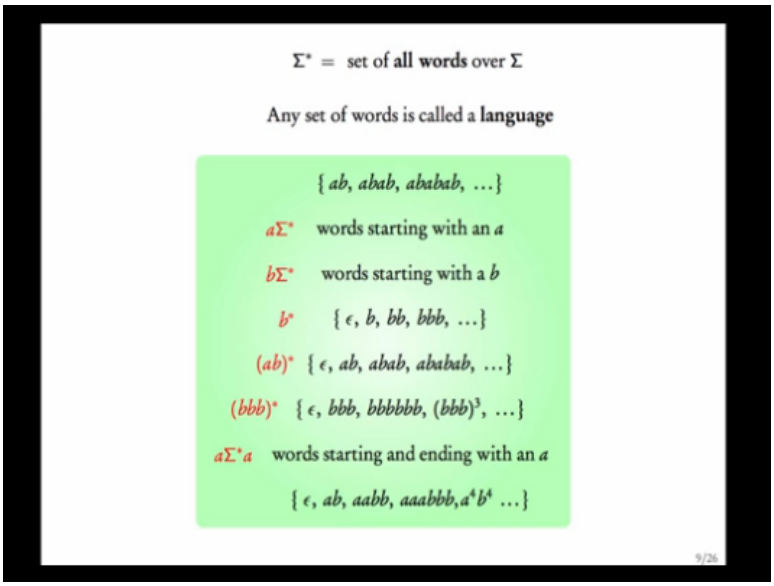
We will define what is called the empty word epsilon and we will see what it means. It is like the one in the integers whenever you take a product of a number with one you get back the same number for example 5 into 1 is just 5. Similarly, you take any word and concatenated with epsilon you will just get the same word. You concatenated on whichever side you want it doesn't matter so epsilon dot bbb will just be bbb.

Any word so w is a word over a and b, w dot epsilon will be w epsilon dot w will be w again. We know that 2 power 0, 3 power 0, 4 power 0 these all define to be 1 similarly we define sigma power 0 to be just the set containing the empty word. Okay, so sigma power 0 is the set containing the word epsilon and this word epsilon is called the empty word and it has length 0.

This is just a technical thing do not bother much if you do not get the point of this. Now, what is the union of all of these; if you take the union of all of these you will get all the finite length words and this set of all words is denoted as sigma star. So, given an alphabet sigma; sigma star will be the set of all words.

In this entire slide the 2 things that you need to look at are; if my alphabet is this then we denote sigma star to be the set of all finite length words and this set also includes epsilon. This is just a terminology there is nothing profound about this.

(Refer Slide Time: 19:29)



So, hence forth we will be using sigma star to represent the set of all words over sigma. Any set of words is called a “Language”. Examples look at this set which we saw before ab, abab, ababab and so on this set of words I am just picking a set of words and such a set is called Language. Another example of set of words, word starting with an a, words starting with a b this is an another language.

Here there is a language with starts with epsilon and has words that has only b’s this is the same language along with epsilon. This is the language containing epsilon the empty word the word bbb and all multiples of bbb. So, if you write bbb cube it just means bbbbbb; this just a short notation. This reduces my effort of telling b’s again and again and that’s why we write bbb cube.

Words starting and ending with an a this is yet other language this is a language; this is the interesting language which will encountered again. This is the language that contains some number of a’s followed by the same number of b’s; a2, b2 a power 3, b power 3, a power 4, b power and so on. These are just languages you can define any set of words and then that will be called a language.

Let us now use this star notation to write some of these languages in a more neat way. Let us take this words starting with an a, How do they look like? You take this set and concatenate it with a on the left, what do you get you get a sigma star. If you look at this set this will look like a followed by any word and this is essentially describing this set. What about this language this can be denoted as b sigma star so you are taking the set sigma star and then you are concatenating with b on the left.

So, for every word here you will get b that word, b the next word and so on; so this will yield a language where every word starts with the b. What about this? If we talk about sigma star that will include all words of finite link suppose I restrict sigma to just b what will I get? I will just get this language so instead of sigma where sigma is a b replace it with b, b star will include epsilon just b then b dot b, b dot b dot b and so on. Okay, replace sigma with b you will get this set think about what is a possible notation for this set using the same ideas. Here instead of b we have a b so we can just write it as a b star.

Similarly, what about this language instead b we have bb that is the only difference so this is just bbb star. What about this words starting and ending with and a? This is going to be a sigma star a you take set of all words you concatenate with a on both sides you will get a smaller set and the smaller set will contain the word that starts and ends with an a.

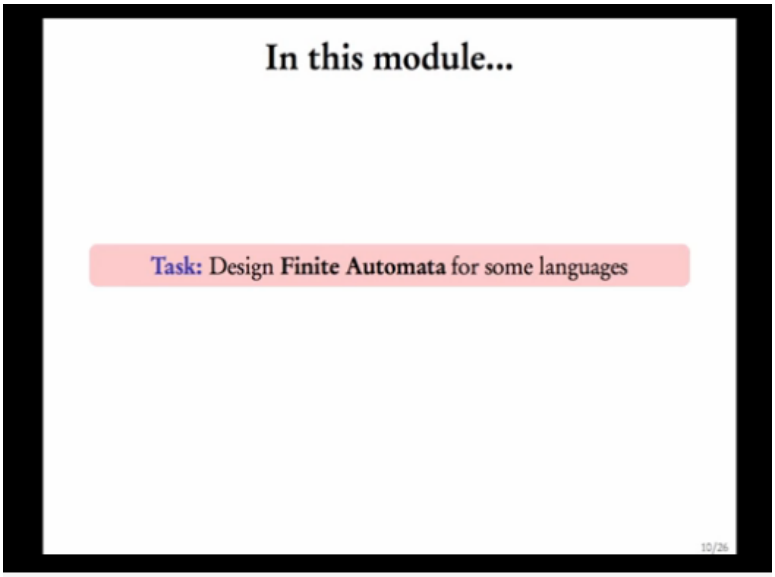
Please have a look at this it's a simple concept that just uses the star notation. People familiar with regular expressions would have recognize that these are nothing but regular expressions but for this course we do not know have to care about what regular expressions are all that you need to know is that sigma star is union of sigma power 0, sigma power 1, sigma power 2 and sigma power k so on.

By just understanding this notation clearly you will be able to understand these notation as well. What about this language? First of all note that this is not ab star or this is not a star b star. I will give this as an exercise try to think why this language is not just a star b star; what is a star b

star? a star would be epsilon a aa aaa and so on; b star is this and when you concatenate this I claim that you do not get this set you get something else.

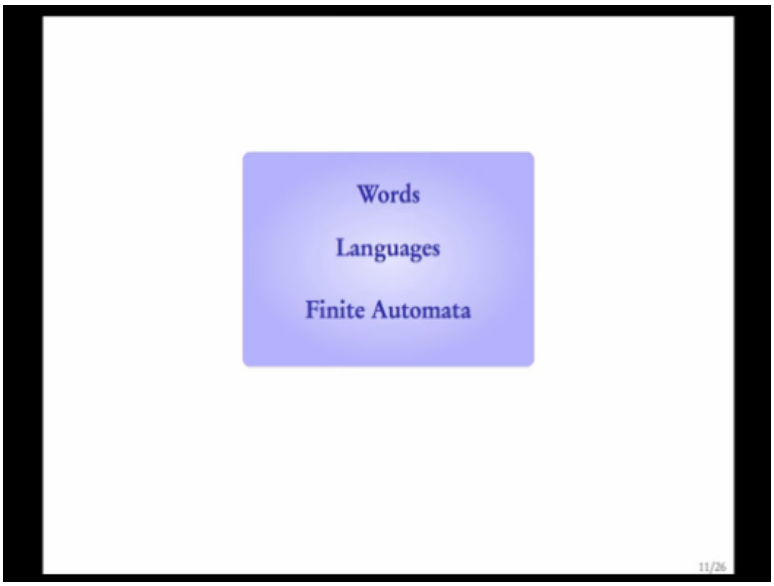
You can have look at it but the point of the slide is to let you know this terminology called language and some notation that uses the star for describing some of these languages.

(Refer Slide Time: 26:03)



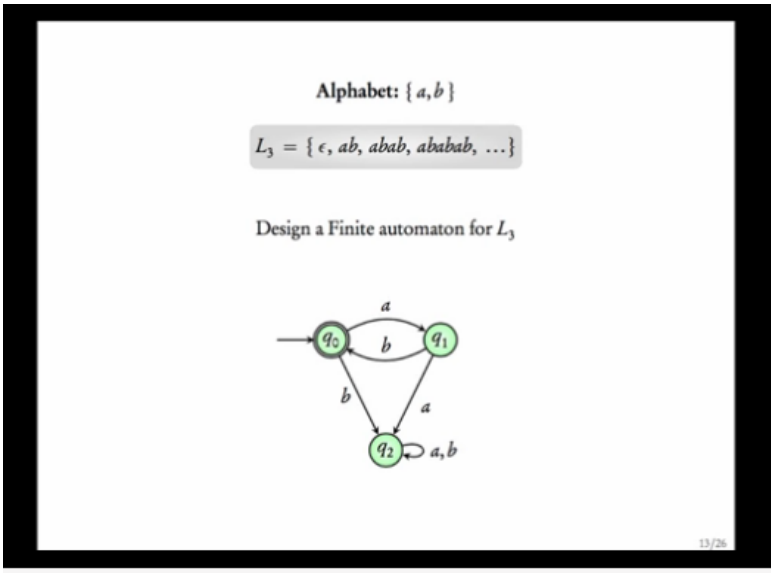
The goal of this module is to design finite automata for certain languages.

(Refer Slide Time: 26:17)



At this point we have take an alphabet you know what words are; you know what languages are and we have also seen an idea of finite automata. In the rest of this module we will give lots of examples of designing finite automata for certain languages. Let's take the language that we already saw instead of the 2 line descriptions of what we wanted; we will just write design a finite automata for L1 this was the finite automata.

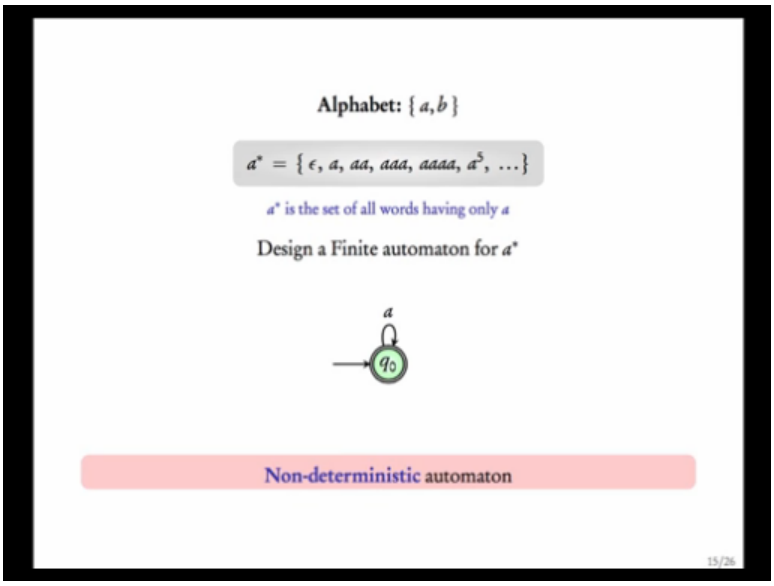
(Refer Slide Time: 27:01)



Let me now modify L1 slightly by adding epsilon to that language how does it modify the automata? Since epsilon is already there the initial state should be made accepting. If the initial state is accepting then epsilon gets in we can even reduce the number of states now. Here we had to go to a b and then start accepting however now since epsilon is already accepting we can reduce it to 3 states each time you see a and b you have come to q0; whenever you see something apart from a b you need to go to a sinking state. This kind of a state from where you can never go to other states can be called a sink state.

So, what q0 if I see a b I will go here, if I see an a I will go here, if I see a b I need to accept so I will come back to q0. However, if I see an a I should go to the sink state if you see the set of parts from the initial to the accepting state will be a b, a b, a a b and so this automaton corresponds to that the language a b star. The only point you need to notice that the initial state is made final to accept epsilon.

(Refer Slide Time: 28:46)



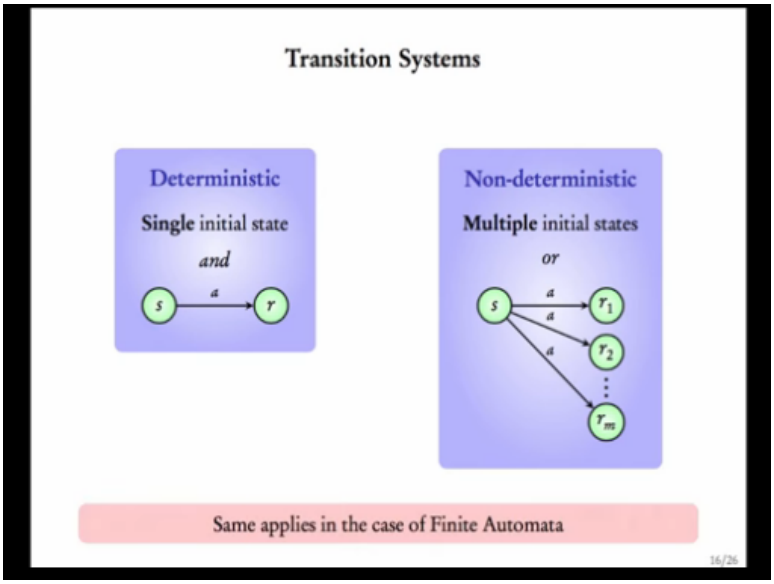
Now, let us look at the language sigma star itself; sigma star is a language. Can we design a finite automaton for it? Yes, it has a single state and that state is accepting you keep reading any letter you will stay inside this language and for any word in this language you can do a reading. So, this just means when I write a b this is the point that I need to say, when I write a b this is the short form for 2 transitions; the first transition with a and second transition with b.

So, whenever I am taking a transition a it is this whenever I take a transition b it will be other transition for b it's just return in this short form saying that there is transition from q0 to q0 on an a as well as on a b and this automaton represents sigma star. Next example, instead of sigma star suppose I talk about a star how will finite automaton change the alphabets a and b. Very simple from q0 you do not let a transition on b so you say that I am in the initial state and the only letter that I can read is a. If I see a b the automaton gets stuck in fact at q0 it cannot see a b.

The other way of designing an automaton would be whenever you see a b you go to a sink state and never come back that is also a possibility but I wanted to say that this achieves the same goal of putting a b and adding a sink state. Something that you need to notice about this automaton is that a q0 on a b there is no transition. Such an automaton is called Non-deterministic.

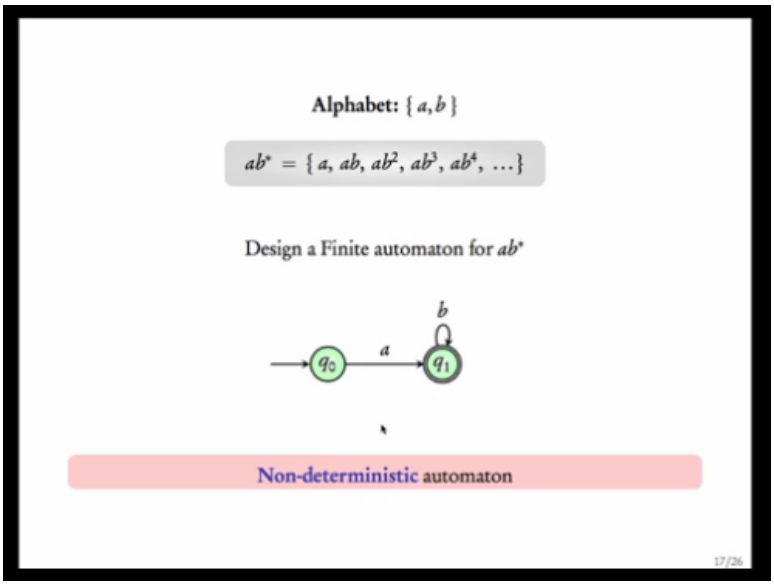
Here the automaton was deterministic at every state on a letter you had a unique choice to take you knew where to go, from q1 on an a you know that you have to go here, from q1 on a b you knew that you have to go here and so on. However, here at q0 on a b you do not know what to do but still the automaton accepts this language this is the language of this automaton. An automaton with this feature is called Non deterministic.

(Refer Slide Time: 31:57)



We have seen what deterministic and non deterministic transition system are in a previous unit. A transition system is set to be deterministic if there is a single initial state and for every state on an action there is a unique transition. It is non-deterministic if one of these conditions do not hold either there are multiple initial states or there are multiple transition on an a or no transition this thing can also include no transition on an a the same structure applies in the case of finite automata as well.

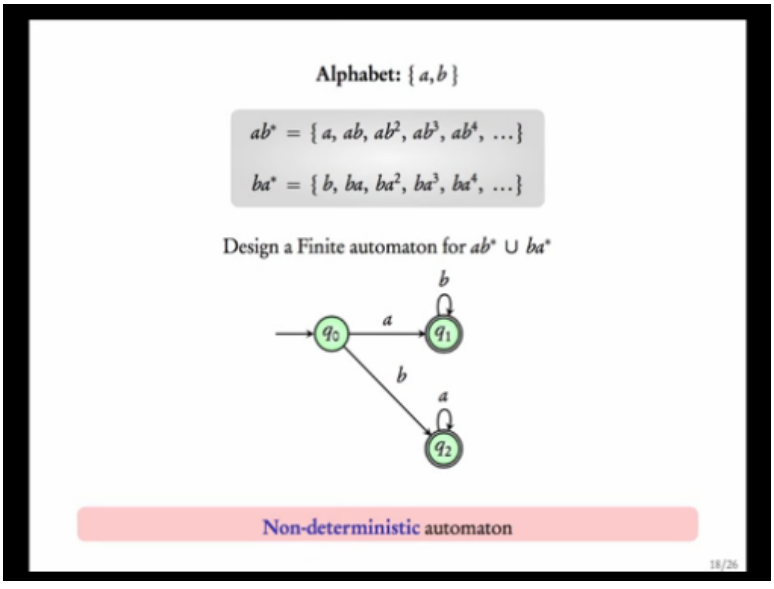
(Refer Slide Time: 32:42)



Now, let us consider the language a b start my automaton should read an a once it reads an a it should just keep reading b. Here it is, this is my initial state I read an a and I go to an accepting state that's fine because a is there in the language. Once I am here I can read only b this is once again non-deterministic because we do not say what this automaton has do at q1 if it reads an a.

We do not specify that explicitly at q0 on a b we do not specify explicitly what we can do we can just make it deterministic by adding this extra sink state and q0 on b goes here, q1 on a goes here but this achieves the same thing.

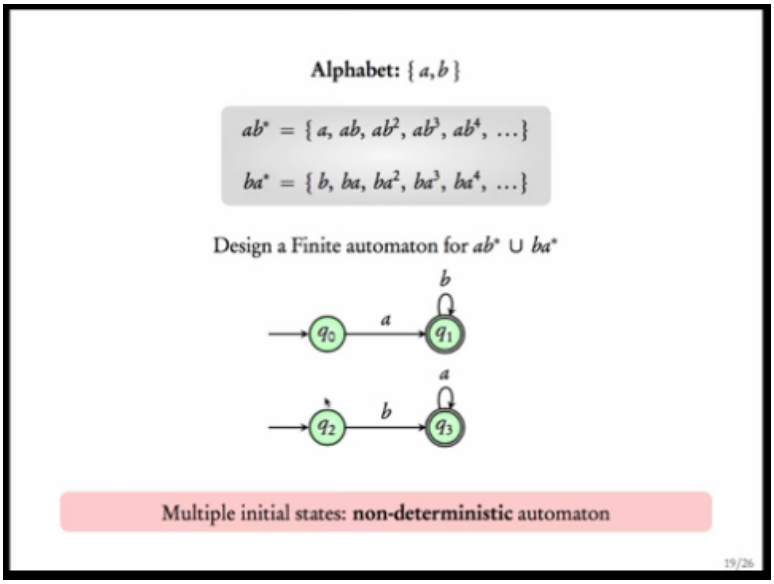
(Refer Slide Time: 33:44)



Next example a b star is this language b a star is the language that has these words b ba ba square so on. Now, I consider the union of these 2 languages that will be another language, the goal is to design a finite automaton for the union of these 2. If my automaton reads a first it should keep reading only b, if it reads a b first it should then only an a.

I am just translating whatever I said into this automaton; in the initial state if the automaton reads an a its should goes to an accepting state and keep reading only b. On the other hand if it reads a b its should go to a state which is accepting and keep reading only on a. Yet again this is a non-deterministic automaton each time we see an automaton I will specify if it is non-deterministic or not.

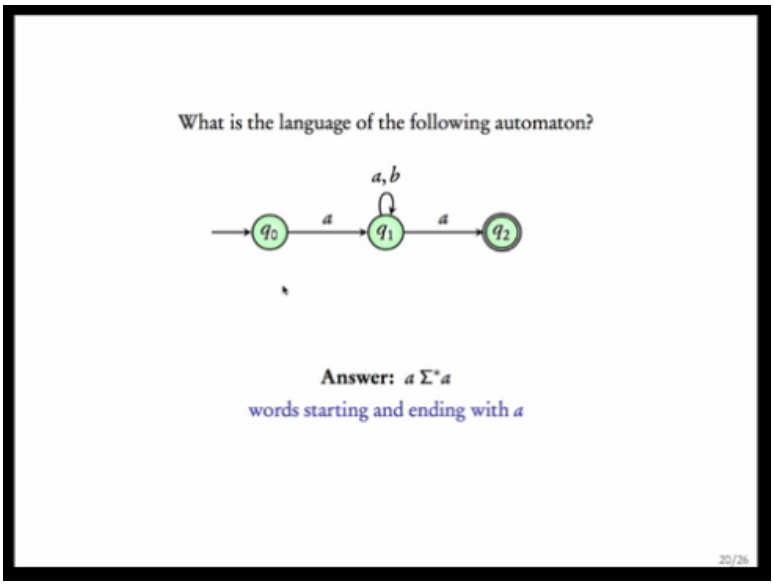
(Refer Slide Time: 34:59)



Let us look at the same language and construct a different automaton for this. This entire thing with 4 states is 1 automaton; however, here we have 2 initial states; either you can start from here or you can start from here. If you start from here you will read ab star, if you start from here you will read b a star.

I am giving this example just to illustrate the fact that an automaton can have multiple initial states and hence this is a non-deterministic automaton but the point to notice that the language of this automaton is still a b star union b a star you have a choice in beginning itself.

(Refer Slide Time: 35:56)

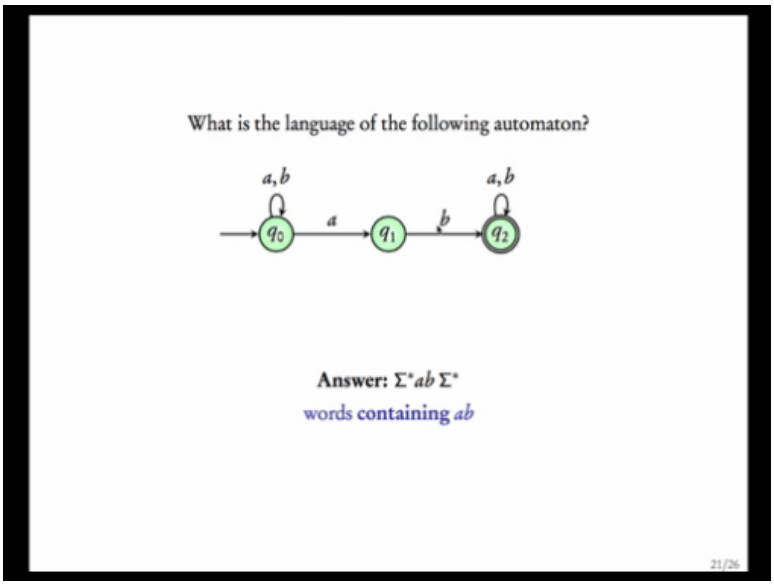


Now, the rest of the exercises will be the reverse way I will give you an automaton and then try to figure out what the language of the automaton is. This is an automaton this is non-deterministic due to multiple reasons first of all from q0 there is no transition on a b from q2 there is no transition at all and from q1 on an a you can either stay in q1 or you can go to q2.

What do you think would be the language accepted by this automaton? You need to look at the paths that take you from an initial state to an accepting state. How do the paths look like? They start with an a read some word and they end with an a. Hence, the language of this automaton is going to be exactly the set of words starting and ending with an a and this is the short notation for it a sigma star a.

Note that a is also a word and here since sigma star contains epsilon; a dot epsilon dot a will give you a. These are some tidbits which show you why epsilon was added if you get it well and good if not you just understand that this automaton recognizes the words get starts with and end with an a.

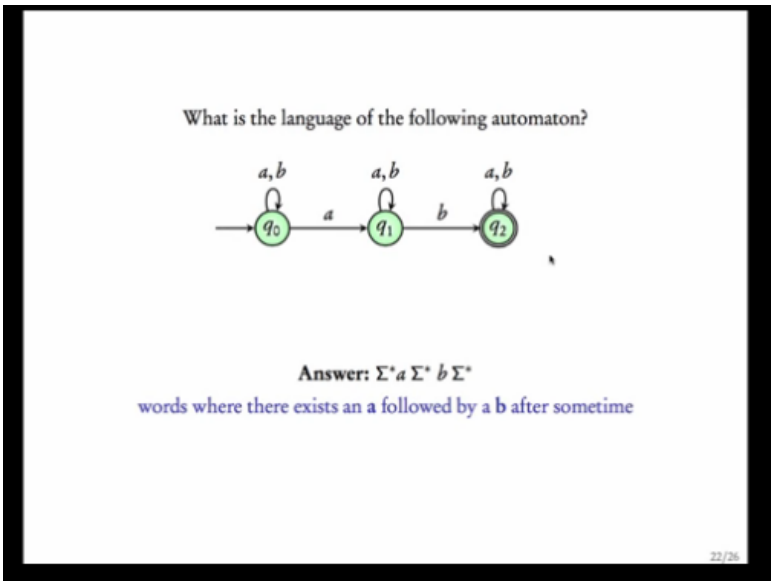
(Refer Slide Time: 37:43)



Now what is the language of the following automaton? It has 3 states you can keep looping here; note that this is non deterministic because at q0 on a you can choose to stay here or you can go here. Once you are here you have to read a b and then go to an accepting state and read whatever you want. So, essentially this automaton accepts words that contain this pattern ab, ab was just indicative you can put any other thing here and such automata are useful to specify properties.

This says that I want to filter out the word that have this pattern a b essentially sigma star a b sigma star; a could be a representative for say red light and then b could be a representative for yellow light and so on. Any way as of now we are looking at it in abstraction where a and b is a generic alphabet and this automaton will give us words containing ab.

(Refer Slide Time: 39:02)

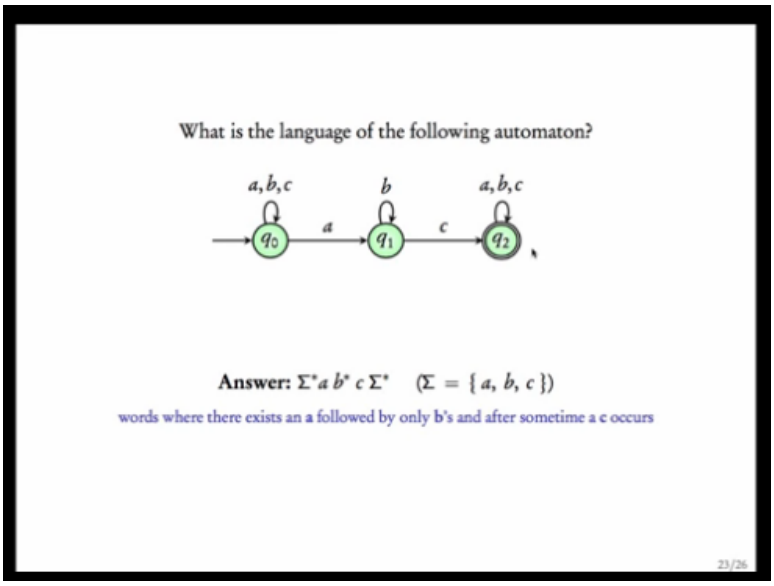


Now, I have modified slightly I have added a loop in q1 aspect so what can be automaton do. It can loop here for some time it reads some words then read an a, read some word read a b and then keep reading some other word. This automaton accepts words get that contain and a followed by a b after sometime for example this automaton can never accept a word that consists only of b's if it had been just bbbb my automaton would have stayed in q0 and it is not accepting so bbbb would not be accepted by this automaton.

However, if I had say bb a bbb bb a bbb it accepts there are multiple ways of accepting bb a bbb but as long as there is 1 path that takes me to an accepting state I am done. This can be thought of say request and response whenever there is a request at some point of time there should be a response that is this automaton. We can also design a deterministic automaton for this so what would you do for determinism at q0 on an a you go here on a b you add a sink state and then stay there at q1 as long as you see an a you stay here.

When you see a b you go to the next state and then you keep reading ab so you can easily give a determinism automaton for this as well.

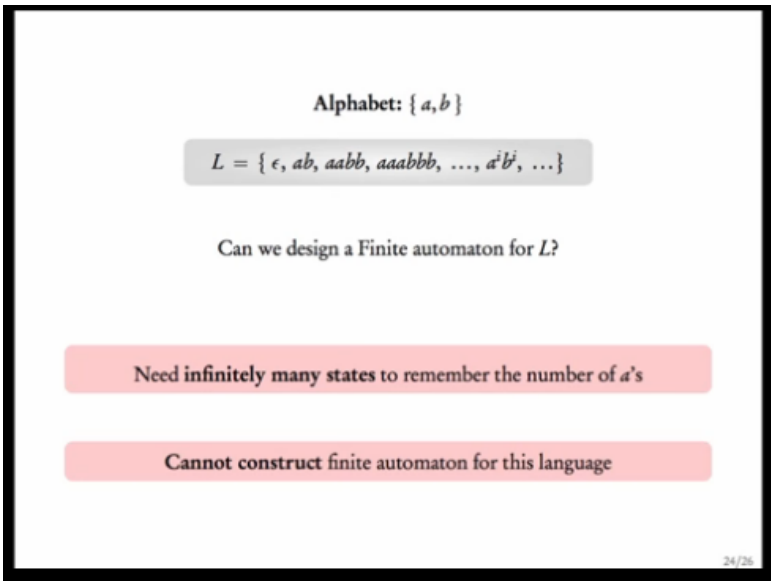
(Refer Slide Time: 40:59)



Now, we have an automaton what the alphabet a b c there is no necessity that the alphabet should be just a b you can have finite number of letters and what does this automaton recognize, what is the language you read a word in a b c then you have to read an a and read only b's read a c and then read any word.

Essentially, words were there exists an a followed by only b's and after sometime a c occurs so this is sigma star a b star c. I have written these things in English just to explain that b's are like requirements in a model something should happen followed by only something else and then something else should happen we are looking at it in an abstract sense.

(Refer Slide Time: 42:07)

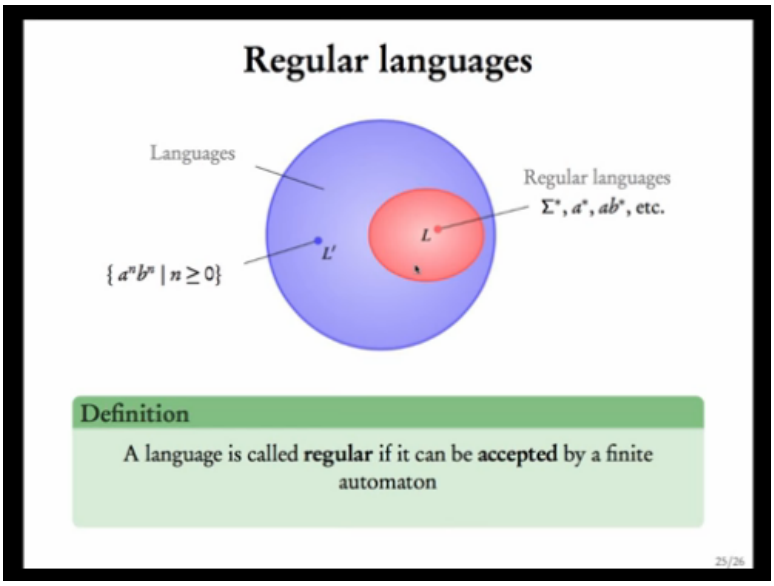


Now, this is the interesting language that I was talking about this language contains epsilon ab aabb aaabbb and so on. It has to see an equal number of b's and a's; it has to accept a word only if the number of a's and the number of b's are the same and more over all the a's should occurred in the front and all the b's should occurred after that. Let me say that if you want to accept this language your automaton should have the capacity to remember the number of a's it that it has red.

If it does not have this power of remembering the number of a's that it has red it cannot hope to read the same number of b's in the future and since the number of a's could be infinite; for example the numbers of a's could be either 3 or 4 or 5 or 6 six and so on. You need infinitely many states to remember this fact and hence one cannot construct a finite automaton for this language. This is not a proof this is just an intuition there are proper proofs to say that we cannot design a finite automaton for this language however we do not have to bother about that.

The point to note is that there are languages for which you cannot construct finite automata.

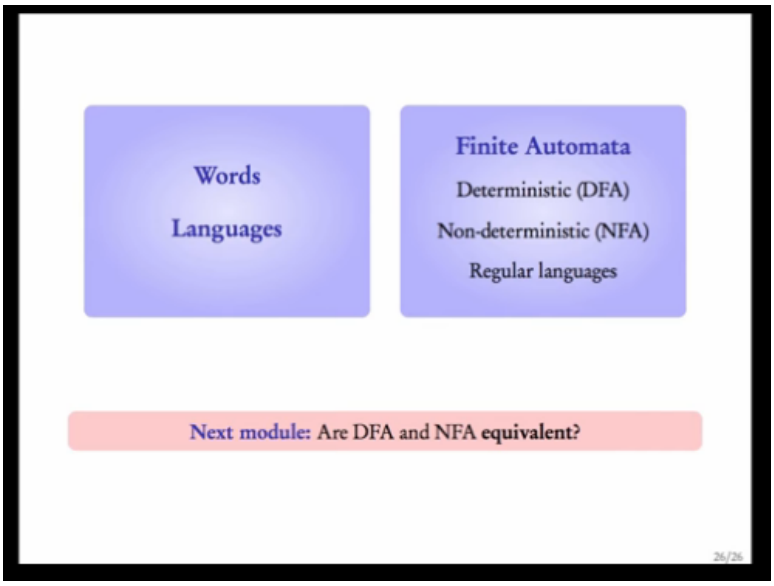
(Refer Slide Time: 43:50)



Hence, we can make this definition look at this thing suppose this is the set of all languages he blue bubble is set of the all languages let this red bubble be languages like this. A language is set to be regular if it can be accepted by a finite automaton this is the definition we are giving. So, languages like sigma star, a star, ab star which had finite automata are called regular. Languages like this which had the same number of a's and b's lie outside the regular languages they are called non regular languages.

For this course we would not care about this non regular part we would care about regular languages. Slowly, we will understand the title of this unit regular properties.

(Refer Slide Time: 45:10)



This brings us to the end of this gentle introduction to finite automata. We saw words, sets of words are languages and for some languages you can design finite automata. A word is set to be accepted by an automaton if there exists a path from the initial state to an accepting state that reads this words. We had different versions of automata deterministic and non-deterministic. In a deterministic automaton for every state on a letter there is a unique transition so for every word there will be a unique path that is traced. In a non-deterministic automaton for every word there could be multiple paths but if there is some path that takes me to an accepting stage I am done.

Essentially, given a non-deterministic finite automata you look at paths that take you from an initial state to an accepting state. This set of paths should read words in the language that we want and every word in that language should have a corresponding path from an initial state to a final state. We have seen a lot of examples you can try to understand this concept by going through these examples once again. We have also seen that there are languages for which you cannot construct automata and such languages are called non regular languages. So, the languages for which you can construct automata are regular languages this is just a definition.

In the next module we will see the link between determinism and non-determinism are they equivalent we will make the notion of equivalence more precise in the next module