

**Indian Institute of Technology
Kanpur
NP – TEL
National Programme
On
Technology Enhanced Learning
Course Title
Compiler Design
Lecture – 6**

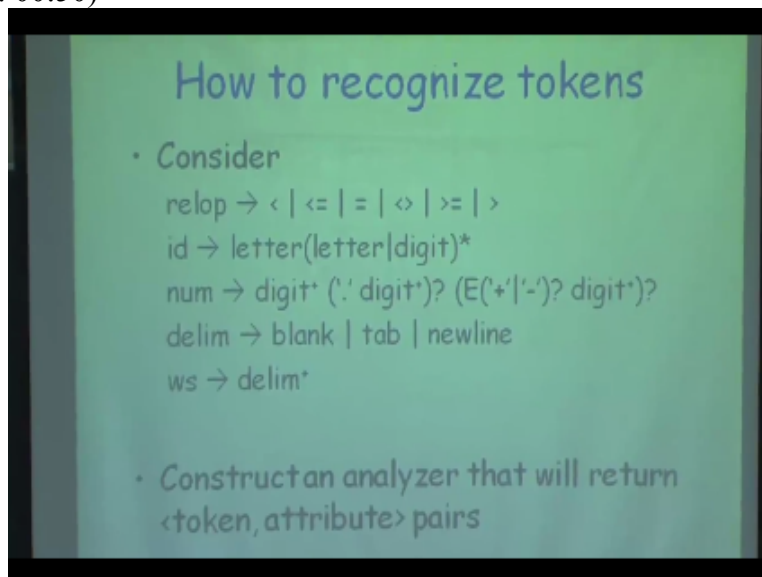
by...

Prof. S. K. Aggarwal.

Dept. of Computer Science and Engineering.

Okay so let us start the capture the previous class we start with looking at specification of redolent and then picked up small language for this the time to costarred. And set the diagram which falling consist please note that two time of diagram. Because the promise diagram is looking at imitation with is. And this was the language we had it is going to now constricted any land which is going to return finally at token at here.

(Refer Slide Time: 00:50)

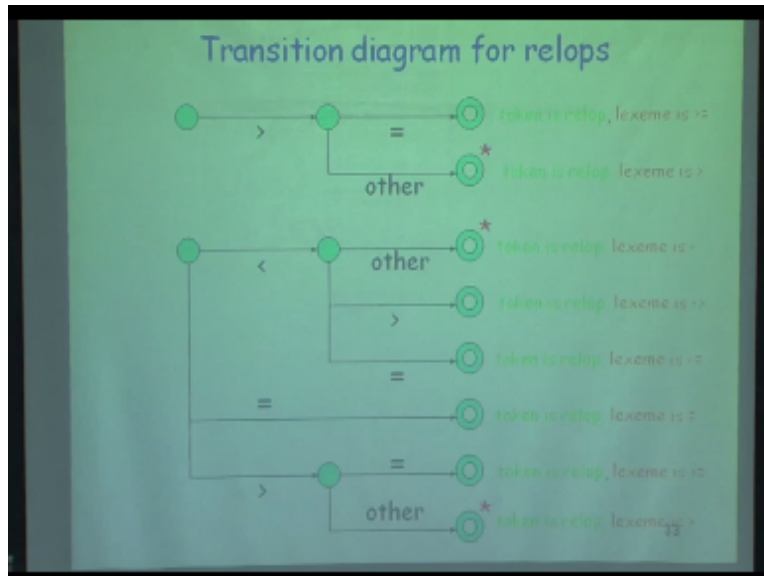


How to recognize tokens

- Consider
 - relop $\rightarrow < | <= | = | <> | >= | >$
 - id $\rightarrow \text{letter}(\text{letter}|\text{digit})^*$
 - num $\rightarrow \text{digit}^* ('.' \text{digit}^*)? (E ('+' | '-')? \text{digit}^*)?$
 - delim $\rightarrow \text{blank} | \text{tab} | \text{newline}$
 - ws $\rightarrow \text{delim}^*$
- Construct an analyzer that will return $\langle \text{token}, \text{attribute} \rangle$ pairs

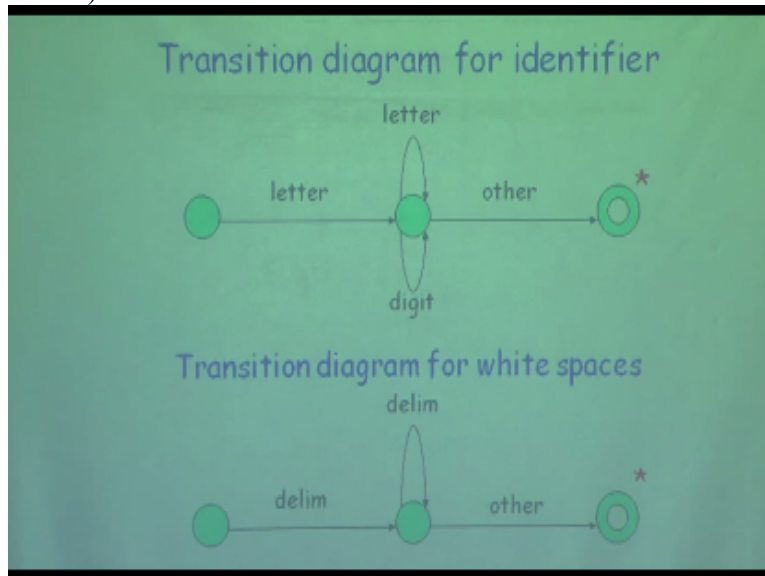
So tokens going to be here either could be relation of token here so it is going to be relation operation with let us see number. And then I would distribute this.

(Refer Slide Time: 01:04)



We made a transition diagram for greater than equal and greater than we look at short sketch of insert by see there grater than symbol for another state and the either symbol to equal other. And this particular state I would return the character back into the inputs. And once we give this and we also started the consists diagram which could capture the whole of operations. This is all the six operators here and this transition diagram consisted of many more final states. Infact six because he had six lexeme here the sum of lexeme we have return here so conclude the number is to moreover each of final statement with some of token with live other than can put it to correct active it okay. This we are start with the previous class any questions or any commends or any doubt? You have before I movement. All this clear to everyone, Okay so let us take little more all states then look at the identify it also I think I am not sure where there I discuss so let us look at this the start state you first look at the letter. And this state look at either a letter are a degree if other the specification set that we consists of defy or negatfy with consists only of so let us look these specification first. The specification are we set of relation operator negative the consists of letter. Forward by zero of relop of letter. Then we have number consists of at least one digit and then we have a fraction part and if you have a fraction part either fraction part is completely missing which is optional which is given by this which you are not but if it is not completely missing. That must have at least a dog and at least one digit but I do note all numbers like one point I must one digit coming off the top and then I have rated the exponent now exponents is either completely missing or it is not worth what if it is there then I have a Δ , T here and then I have an optional sign which is either a plus or minus so even closer - could be missing and then followed

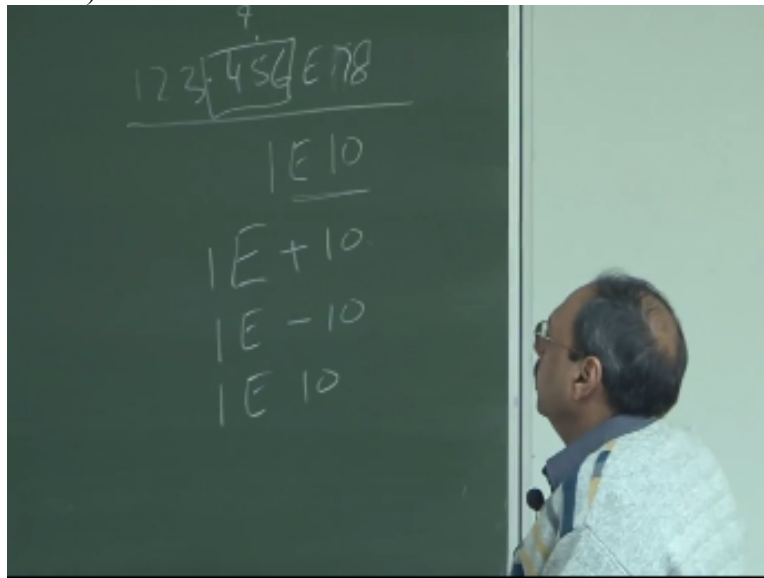
by at least one digit okay so these are my specifications and either blank tab or a new line and the white space is nothing but one or more very specification something. So we looked at how to construct a finite how to construct a transition diagram for relational operations and then we have to see that from the rest of the specifications will then go for implementation of these transition diagrams if that is the fastest front of us so this is the transition diagram for relational operation and then what we have this transition diagram for, (Refer Slide Time: 04:17)



Identifiers so I first be letter then I each another state and in this state I will be in number of vector for the X and if I see something else then I reach a final state which gives me a recognition of identifier but it also says that I must return the last character typing to the image and then we are looking at coefficient on Y spaces so this is really simple I must be at least one delimiter and then I can see more than limit as well I can have any number of blanks here and when I see something else when I eat the final state and I determine the last character that either by specifically likes.

Look at how numbers okay, Now what I am going to do is I am going to actually construct multiple completion diagram pronouns okay and how do these look how do they look so in the start State okay no matter if I am looking at number I must see a digit so this is what we really do that I will actually see a digital task not dependent upon set up this is in India or this is a real number which means it has a fraction part I will either see a dot or I not 0 right. So but before I reach that stage it is possible that I will have more than one negative so in this stage I said I can loop on the gate and I can see any number and how do I exit from this so if it is a real number then I can only wear it and exit is why you think but there is another way I can exit

from this because section part will be missing that is the optional part and I will directly see an exponent part so I can have something like this one keep that that is possible right. So what I do here is so let us leave this part and let us see that if I have seen that a dot then I must see at least one degree so this is what it is one you get gives me I use distinct and in this stage I can see more dangers okay so I was I can actually move on the particular state in this now once I have exhausted all so if I look at now let us say.
 (Refer Slide Time: 06:32)



Something like so by this specification more tired exhausted is everything up to physics so in the spark state I will see one, and then the next day I will see two and three a loop on that state then I see the dot and then four they peek through this state and then five and six will loop on this right and then in this state now either I can see an e here which means I am looking at this or I can see an e here because this part was completely.

Optional it so in this stage I can either see any and reach this tip or I can see an E and H this state and if I reach this state then what do I see from my specifications I can either see a plus or minus or that is optional I guess be a video yeah so this is what may happen that is see if those are - but I may see a leave after plus or minus or minus C and it immediately so I am looking at situation then I say that I can either have $1E + 10$ or I can have $1E - 10$ or I can $1E 10$ which in this case is by default same as interpretation.

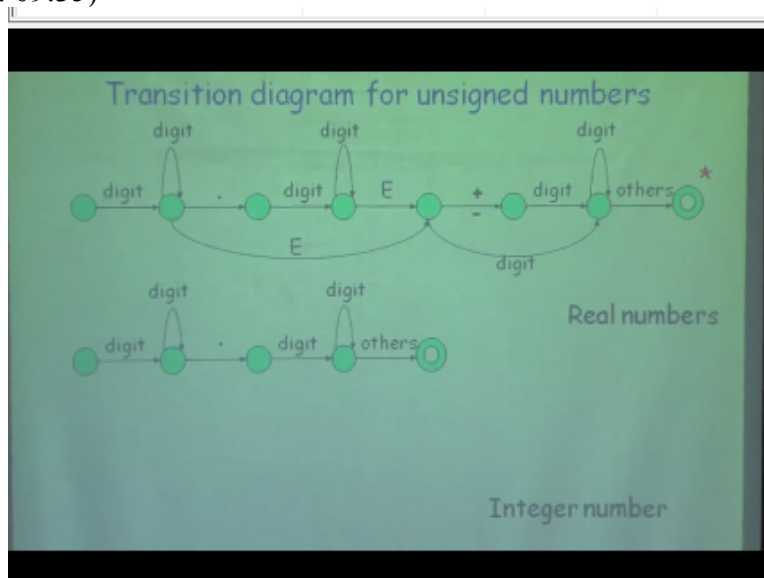
So I have reached this state now I am not going to count yes and therefore I say that I will keep to hear more and more digits okay. It and then we like the mouse when I see something else so when I see something else I reach the final state in this final state I what I have not written here and which is going to be part of the construction of the number so if you recall the code I hope in

C when I said when you read a ticket then how do you construct the number keep on reading more and more digits keep on multiplying.

Whatever you are seen by so are by 10 and already so if I see one I take that value of 1 and subtract as a value 0 this give me one and when I see two I multiply one by ten and difference of this character to with us Eva and so on so that way I can construct this number having similarly you can have the logic for the fraction part in context current that is no big so this is one proposition diagram you see for unsigned numbers let us look at other kind of transition diagram

in now suppose I just consider situations like this,

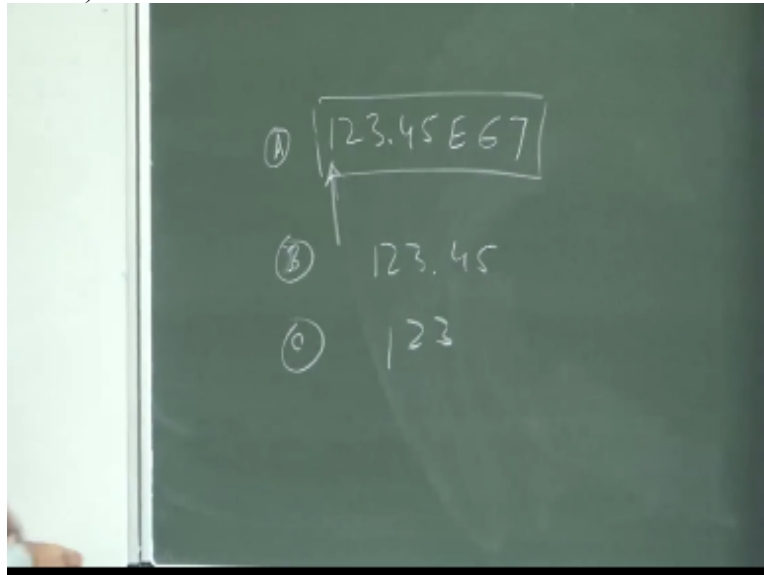
(Refer Slide Time: 09:35)



I do not have an exponent okay so what is going to happen if I can so this is when we going to be a sub diagram so I not go through the detailed application but you can see that here I have the left hand side of the number third and here I have the fraction part and then I will exit when I see something else so this will be captured so this is actually going to capture real numbers which do not have exponent passed this captures real numbers.

Which have an exponent part and then I can also look at just the integers and just the images are going to capture you should not have either a fraction or are not going to have exponent part and then in this case when I see others I reach the final state and then I return the last character back into the industry so this is the possible set of transition diagrams for capturing all possible numbers now I have a little problem and the problem is there are three transition diagrams which have start state and I have a number so suppose I want to now say that I have organized this number.

Now which start state I should start it because these are all corresponding to the numbers and if I just keep the first to you that does not tell me what is going to follow this is a real number mother exponents while various combinations and just by looking at the first video I cannot make out what will be my star see so what should I do so suppose I am trying to recognize, (Refer Slide Time: 11:00)



And I have seen this what should be much tougher should it be this so what one standard and 115 statistics that I should prioritize and then if it does not succeed then I should go to other now what should be my order in which I need about two parts or tokenize this particular string let us say I give priority to Atria your chair and I get a new one two and three and then I see about and I say I have reached the final state so what happens is that correctly no because then you say I have tokenized this five this is the beginning of another token.

Because this will be returning to the feasting and then you will try to construct another token starting with this which is obviously give me an error your input is incorrect okay so that's priority order that India comes first is not correct now remember that you are using maximal invincible by using that principle which is the transition diagram which is going to consume the maximum input first of all of this so that should be my order so what we should do is we should prioritize these routes but we should not be a final failed state.

Unless I have tried all those so let us look at that scenario suppose I have I give you three scenarios this is one of the books I have this is now let us see what happens in these three bases so when I try to tokenize this it will actually fit here and we reach final state and we say this is the top but now suppose I try to organize this by looking at the first diagram because that is my

minus 5 okay so this will consume 1 decisions unique decision you know dot is equal to the for the second room right.

And now what happens I am expecting an e here this Louie okay so I said this is an F but is this really in error then say no this is my second position diagram let's try this and when I try this will successfully each a fine what about 123 will take me to these snakes 1, 2 and 3 and then say I am expecting a Tory and I count go beyond this so this obviously is not the correct term vision bragging then I come to this so I will get 1 2 NZ and I am expecting a dot again I am stuck then I say let us try this one I Phi 1 2 and 3 and then I reach the final state alright.

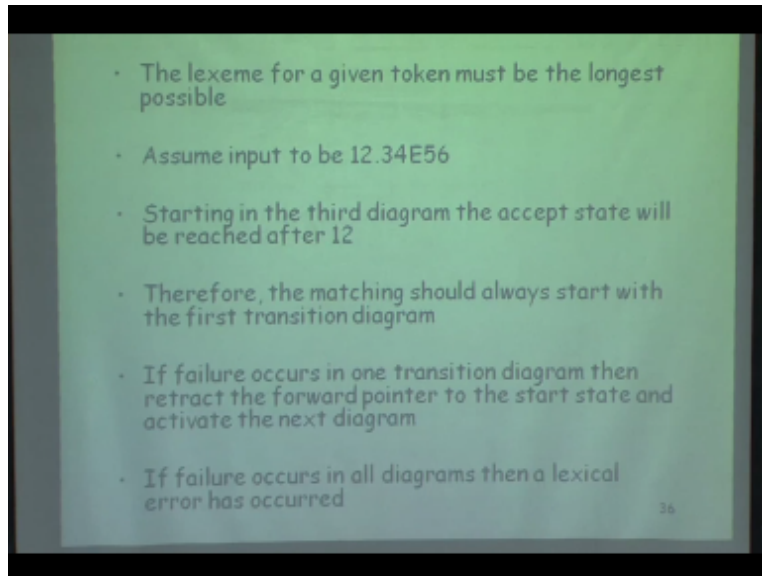
So that seems to 1 so if I could prioritize my rules okay then I will be able to decide doesn't matter which input I choose I can just go over this first list first in this first and only give all the transition diagrams reached the page state one after another in this file today we do not have to there is no 2nd this is the first video went to be out there is anything other than go and eat and take it to the final state and make it we need it record and it this I think if they do thought then take it further.

So what observation is that unnecessarily I know when people transition diagrams I should somehow have a transition diagram where everything will be part of the specification and then when I can do one here and two three and I do not see a lot of e and I see something else that should immediately jump to final state is that what you observation is okay anyone has an answer fire I will do that then we are trying to convert this into number to take point number would be hundred a kidney like this would be anything if we start assuming that the number would be floating by number.

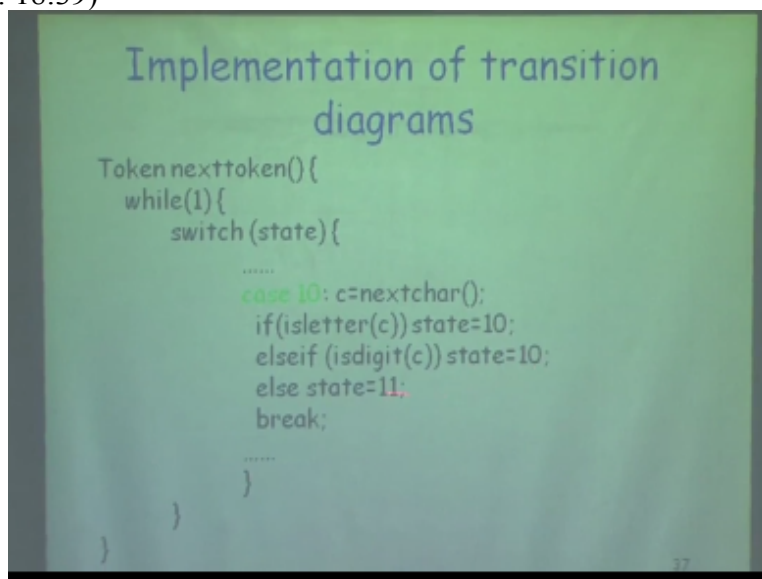
No we do not start will be decided by the final state so if I look at this so look at this I mean water suggested if I say this the news one is convinced routine then I am expecting category and if I say others suppose I have a nine others still take me say finally here then even this is okay fair enough so if I say here that we take me to a final state here okay if I say other so instead of going on this side of this end and come to that which is amounting to the same thing all I am saying is that I am now returning so with me delay answer to think about it let me delete this answer.

For five minutes and then we will come back first let us see how do i implement this because we are talking about implementation and what I have seen here are specifications okay so how about event this.

(Refer Slide Time: 17:53)



Yeah so okay so this is this white actually captures we have already discussed so let us see for a given token must be as long as possible and assume this input 12 point B for exponent 5 6 and starting from her diagram will accept state which is final state after 1 2 which is nearly wrong and their formatting should always start with the first condition diagram which in fact Thomas and if a failure occurs in one condition diagram then detect and then forward pointer to the start State the next condition diagram this is what we are doing is what we discuss and if you create occurs in all diagrams Diana lexical level is a cop otherwise I so this is what I just discussed by way of having these examples. So how do I implement these transition diagrams?
(Refer Slide Time: 18:39)



I can just have a switch statement and all we are saying is that when I look at the next token and I can go into an infinite loop I can keep on reading characters and all I need to do is look at the

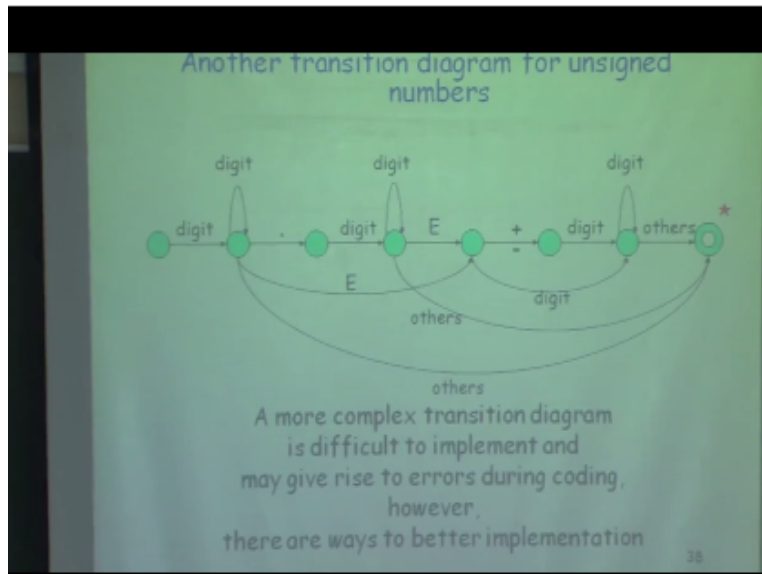
stage look at the input character and then decide what is my next stage and in between I do bookkeeping book then what that we do certain numbers so this is what happens if I am in some state 10 say I have a doom certain state if I am in some state.

10 then what do I do I read the character and if this character is a letter then I say state remains 10 if it's a date then state 2 means then otherwise this is state 11 and then I break so what transition diagram is being captured by this Court which transition that actually get it slightly different thank you so this is saying that I mean such a fake ID is the same state and then if it is something else then I move into this and then I do and you can put up all the bookkeeping information like.

How you are constructing a Lexes and how you are returning character back into the input stream and so on but basically this remains the skeleton of activities so it is very easy to take a set of foundation diagrams and just convert them into a piece of seafood so remember we talked about three implementation standards one implementation strategy was ready write regular expressions and like some fool to the generation of lexical analyzer second reset was going to be I am going to write certain specifications and in a systematic manner.

I am going that people low-level see and third was that I'm going to use some kind of low level 44 I David I was going to be done by assembly language so that it can be faster now that will be no different than this because the only thing that will happen is that when I spoke about the next character and so on which is actually a read statement that time I am going to use some kind of assembly there because that is going to further improve at least speed of my pool so on transition diagrams can be easily converted into something like this and now coming back to the question which was less saying ideal by one.

(Refer Slide Time: 21:09)



Why do not buy much so it is actually worth it in this state I can see others it can be chair in this state I can also see theirs and if I add few more edges then we are able to complement and this will capture single transition diagram is going to capture all numbers successful okay what the problem is that when I start converting this partition diagram into a pole the more complex transition diagram you have the more complex.

Your code is going to be because that every state will have to make multiple diseases now the choice is there is a trade off and the head of this you want to have a complex transition diagram and therefore complex code and therefore probability of introducing errors or a set of simple transition diagrams and then simple port and hoping that because systematically right in SICU we do not have a pool which is going to make sure that I just write the transition diagram or I just take some kind of regular expressions and convert them into C .

So trade off really is that context Commission diagram again okay now if I start loading everything into a single condition Braga perhaps. It will become very complex but there is no major where I can say that there is no objects and this is the boundary beyond which you should not do it is really a judgment the programmer has to make so if I take this transition diagram put more complex then what we had earlier for some transition diagrams when we you have to examine people transition diagrams you have to remember the pointer right that you say that corresponding two numbers have five transition diagrams.

And this is my start state so if it is VD reaching the failed States then I say that if that is the start state 11 is this horse values the start state of universe then after reaching the failure state jump to fell or jumped to 13 and then will be same so little more messy lost you only have to move your input back pointer back see it is an additional bookkeeping nothing is getting lost anyway

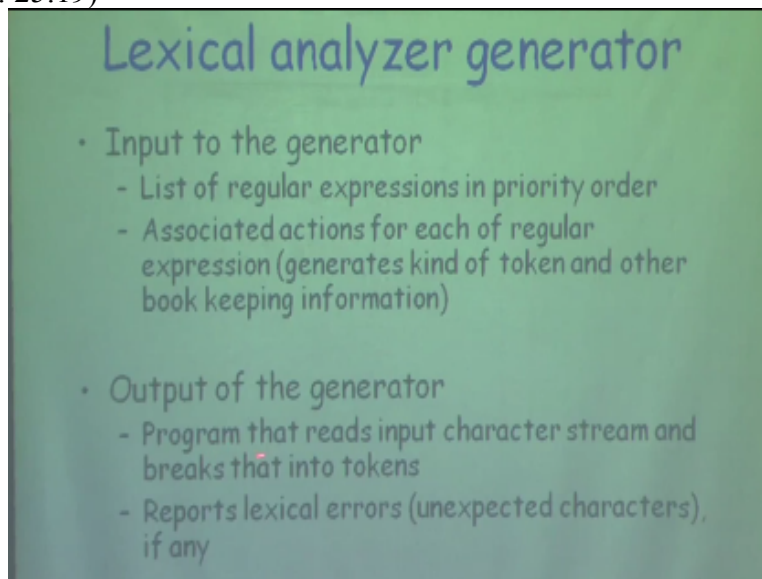
right so when you say I have multiple transition diagram I can have one more pointer to the input and say.

That this pointer will remain right so only catch here is that why did I have not this transition diagram earlier this was the point I wanted to bring out that I can have both the specifications both can be implemented you have to make a choice which one is more complex and which one is easier to me.

That is the only this is like saying that if I ask you to code something if I ask you to solve a problem you can write hundreds of programs which will give me the same solution of which one is right all of them may be right which one is easy to debug which one is readable and so on that is a subjective death what is this point here to everyone okay. So let us then move on and let us see that the third approach okay.

So this is we set specifications first and then I said from specifications how do I design plans transition diagrams and then how do I convert those transition diagrams into direct sequel okay but suppose I do not want to go through that good then what do I do yes.

(Refer Slide Time: 25:19)



The slide has a green background and contains the following text:

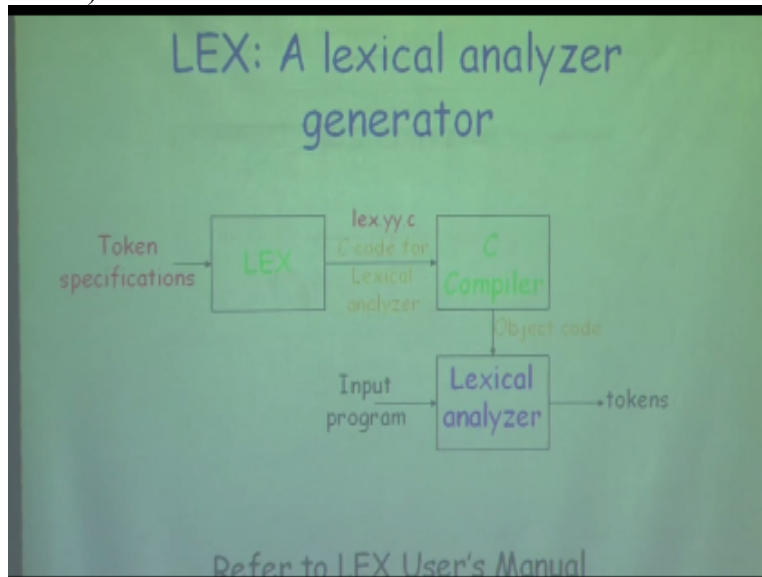
Lexical analyzer generator

- Input to the generator
 - List of regular expressions in priority order
 - Associated actions for each of regular expression (generates kind of token and other book keeping information)
- Output of the generator
 - Program that reads input character stream and breaks that into tokens
 - Reports lexical errors (unexpected characters), if any

I am going to do that use lexical analyzer delicious and you already know what I can I can either generator which is like this yes to five everyone remembers that in fact yes I have already sent to the user let us engage so input to the generator is a list of regular expressions inserted prior yoga and priority order is again going to be very important okay and then we have some sort of section which is associated with each of the regular expressions what kind of evening and once that is done my output is a program that is going to then reach out to the C program which you again compile.

So if you are talking about Lex then output is going to be Lex dot YY dot C which are going to again write the small main function and you are going to call that function left or YY dot C and then compile it and this is going to then break your input into second programs and it also going to report all that.

(Refer Slide Time: 26:18)



So if I look at that particularly this is how it looks that Lex takes a set of inputs which is tokens per second action gives me Lex dot YY dot C which is the code for lexical analyzer and then I have the native C compiler which is going to give me object code and this really is the lexical analyzer which takes my input and gives me a set of codes and this we have so same thing going so for more details just look at the Lawman I am not going to discuss Plex once again if you do not remember something otherwise I mean I expected.

Something you remember the key things to remember in exist most of the things he lacks has not a function calls and gives you lot of so generically but we to remember the very few things most of the time functionally you can write correct programs they may not be very efficient but they at least functional use the right things you have to remember is at some point you need to capture the Lexi okay and how do you capture the Lexi in legs what is the function for that there Is an I think YY Texas right so YY Texas will tell you what in the string we just be matched by a token and by YY line will give you the length of that and only one additional feature which is going to be useful is to look at it.

So if you are looking at context if you recall a slash leave it says that match A only if it occurs at the physics of B right so if you remember these three things and then bye bye in Y by L is level of function that helps you in inputting and that will tell you how good just remember these things

four five things will at least be able to write functionally correct program and I am assuming that you at least know your regular expressions so that you can write specifications there.

So you do not need to know lot of things about Lex now another thing yeah you should always do is look at this file next not bye. C and open it using whatever editor you use and search for a string or debug you search for a string for debug most likely this is going to be align debug which is set to by hash define is set to see take them, change that to one and suddenly you will find that when you start running this program we give a lot of debugging information so both in length and in get if you set the debug switch to one or this variable to 1.

Then we get lot of debugging information which will help you in debugging you this so just go through this code once but the code is not difficult look at e few examples and you will be very efficient with or very comfortable with Lex word, advice is not to change the leg sport or this generated code to manipulate certain tokens all the time you should try to manipulate your specifications and not any manipulate code unless you want to be.

But some in production code it is a bad idea that you first generate some code then go and modify that port because next time you try to modify it to not work at all in fact I mean we want to focus at this part and not modification it is true or false right so I do not know whether we have tertiary logic so if you say to any number other than zero it will still be if 0 is false sending up any other number is false always okay.

So how does lakhs worth it lakhs again this is just to recapture about functionatural planks and to capture about what to do in curing of tempotation when you are trying to construct finite state machines from regular expressions basically what it does is takes regular expressions disparity language and which can be recognized by certain finite state machine.

(Refer Slide Time: 30:21)

How does LEX work?

- Regular expressions describe the languages that can be recognized by finite automata
- Translate each token regular expression into a non deterministic finite automaton (NFA)
- Convert the NFA into an equivalent DFA
- Minimize the DFA to reduce number of states
- Emit code driven by the DFA tables

And this is our standard really that what is construct in non deterministic finite state machine minus ϵ and then convert the same plan NFA into an equivalent DFA and then to minimize the DFA to reduce number of states and this part again you must have done is part of few and again if you forgotten it ,it is good to know about the internal of the toonly attractively used these are the really stacks which are largely logically involved.

Your code may not look like T functions like this code may be all merged into each other but logically this is what it does yeah and the code is different code is iniquitous yes it is so again if you look at Lex dot YY dot C you will find a large table there which is being used table is nothing but a finite solution is telling you that if this is my state and this is my input that what should be the next questions.

In that all about lexical analyzer or one if you have questions We can discuss those questions here otherwise we are going to move on to the next place of compact so in classical analyzer to everyone yes no yes so second temporary the vent angle DS have given so writing Lexical for what about to write yes 2nd February is date so today is what is the date today 22nd only 10 days so you comfortable time in fact you have to be able to finish much earlier my suggestion will be that do not start saying that oh I have ten days and therefore I will start working on 30th night and finish everything by 31st night.

As usual this will be worth so no problem and second will be able to submit because goes to the Teddy's years and I am going to get out of bus saying why I do not understand this obviously for next 10 days if you do not work you do not understand my suggestion will be start now you know everything about lexical analyzer and finish it in two days once you are finished just forget about it review it one day before the submission and that will work fine.

But if you start coding on 31st or 30th it most likely will not be able to finish your lexical analyzer by second so this last lap of marathon kind of situation every night and hoping that you have done very fast in last hundred meters and in the marathon that usually does not happen the constant speed success you really want to finish it success how much do you need to learn about the language this is a very loaded question okay so when you say that I want to learn about language okay what do we mean by saying that learn about the line.

So I can interpret this question in several ways that if I am trying to learn English then I want to learn about how to write a good essay In English what I can say I do not care about that I just need a dictionary and I need to hook up early so if you think that I want to suppose a language coordinator for the language fortune for beneficial for XYZ I do not have to start coding in that language try to compile all I need it to do is I need to look at this specification so the language and read that.

How can i tokenize strings in this particular language that is the only thing I need and nothing else so if I cannot write a program in that language that is fine as long as I understand it in fact now my solution will be which is the first language you learned C or D , C and what was the book you used Manikandan how many section and chapters you looked at so go back once again when you learn to see your how many sections in chapters the data so go back once again open critical energy and you will fight that somewhere in the middle it says veterans man look and the first part says programming in C.

But the second part will say C reference language C reference manual is the part we need to learn which is say how to organize or cook at the gamma what is the semantics and the first part which says how to write programs in C forget program and that is true of all languages so look at the reference manual and not users matter because your references and nothing else so if you say that I am writing say compiler portal that if I cannot code in Portal that is okay.

As long as you know what are the specifications of only thing only language you need to learn is the language in which you are boarding your compile, that is more enough any other questions every one or okay there is another question that we have not meet after an announcement that everyone are comfortable with the broken hand they understood each other know still fights going on putting your so all those who have fights and distuets with your team partners please meet me tomorrow so I can dissolve them so that can be protective as faster as possible that is my job.

They have to come and tell me that why you are not able to work with them because by and large not always I have gone by your choices I am sure that some people have not got their choices but

at least one member in the team if you choose right in most cases you have other team which one of your choices cannot be in the first choice but at least so still you have problems with working with team you have to discuss it now other than that 16 of people standing on the dash and say this can we work I work and we were not that will be really bad.

Ask me these kind of things happen past I am just talking from experience and saying let us try to dissolve those problem now other than final depth okay shall we move on to the next topic if you have no other questions alright so what are the next phrase compiler what are the next phrase of compiler parts okay so good.

(Refer Slide Time: 36:51)

• Check syntax and construct abstract syntax tree

```
if ( b == 0 ) a = b ;
```

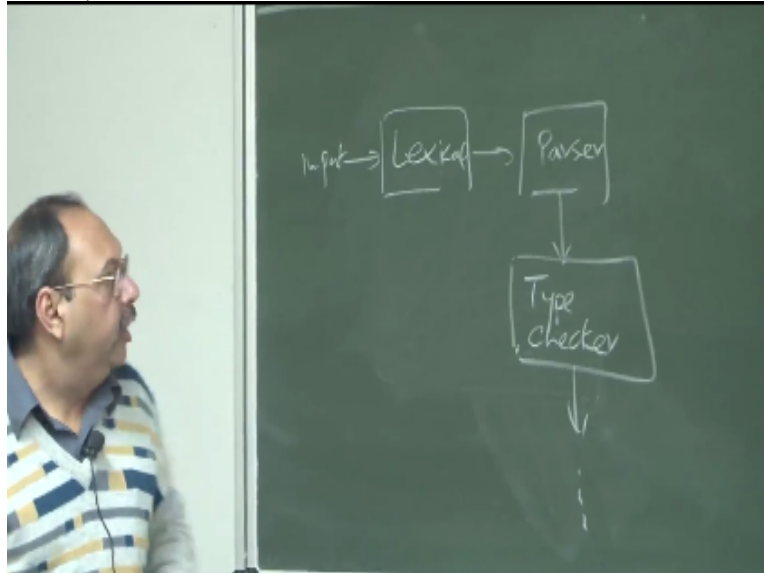
```
graph TD
    if[if] --> eq[==]
    if --> cond[0]
    if --> assign[=]
    if --> stmt[a]
    if --> sem[;]
    eq --> b1[b]
    eq --> 0[0]
    assign --> a[a]
    assign --> b2[b]
```

• Error reporting and recovery
• Model using context free grammars
• Recognize using Push down automata/Table Driven Parsers

This is the noun and in the first five I am just going to capture the functional specification of part setters this is input to my answer my foot have already been brokenized and I have no one have to correct the have a string of tokens for words every one olives so now I have this A FedEx this is and B and = in comparing give and then I have a this is a way and semicolons and output of my parsers is going to be I am going to put abstract index free and I also want to do error reporting that means what to say that if string of tokens does not compare to the grammar I have for the language that I want to flat in error at the same time .

I just 1.2 exit of rining error but I want to recover from this situation and so many errors possible in the process of parsers to recover and we talk about most strategic for a reserve week and for error reporting and from the model we talk about context free grammars and you know that how do we recognized context free grammar and you know is and I also the language and you know that table so this is really what we are going to talk about the next five lectures or so we talk about theory of parsing so but this boy actually captures the complete functionality of

what my parser is going to do and what I am going to now capture in vessel like this is how are you now before I get into these details of parsing let us also understand what is cannot okay so if you see here parser is sitting somewhere in the middle of process of summarization.
(Refer Slide Time: 38:56)



And here what I have is lexical this is my input is this is their parser sits this is where my type checker sits like that okay now 2 things must understand one this concept can do something when but it cannot do what type checker is post okay also parsers can do this lexical cannot do so parsers can do it here and just needs to not only needs to check that my input is correct but also need to generate information for rest of the phases so just by saying that if I have a context free grammar in our specifications and this input term conform to the context free grammar specification Saha then this is wrong input and then stop or says this is right input and then swap is not acceptable.

It must generate enough information so that the phases which come later so they can do something more mean they can do some so first let us see what is in text analyzer can on in now the kind of questions.

(Refer Slide Time: 40:13)

What syntax analysis cannot do!

- To check whether variables are of types on which operations are allowed
- To check whether a variable has been declared before use
- To check whether a variable has been initialized
- These issues will be handled in semantic analysis

And this is a normal situation type of questions we say you have to check whether the values of certain types which operations are allowed so for example if I say $A + B$ and if you say S this operation is allowed on A and B parsers cannot be done here that will be saying that in which context this $+$ operators has that will not be able to put there also we not able to for example check things like whether this very well before I so it will not have to check whether he has been declared before I am using it.

That is really not the job of this page that will be done subsequently I want to check for example whether this variable is initialized or not so I am not be able to say whether this is undefined value and therefore do not use it parsers will not able to do and all this issues are going to be so basically if you look at all these three points in some ways they are talking about context information okay this one is saying that in what context this particular variable occurs and this is also saying that this variable has been whether it is undefined or uninitialized variable and so on and this operators.

And so basically whenever you see some contextual information parsers cannot handle contextual information okay and that is why you say that we have more in this context freedom and also because we are saying that we want to do things here there are kind of things which are not able to do here that would finish In the like lexical analyzers, what are the kind of things we want to do here which are not able to do here which can I can so now you know that in the language hierarchy white context tree grammars are more powerful than the regular expressions because they can do few things regular expressions for example.

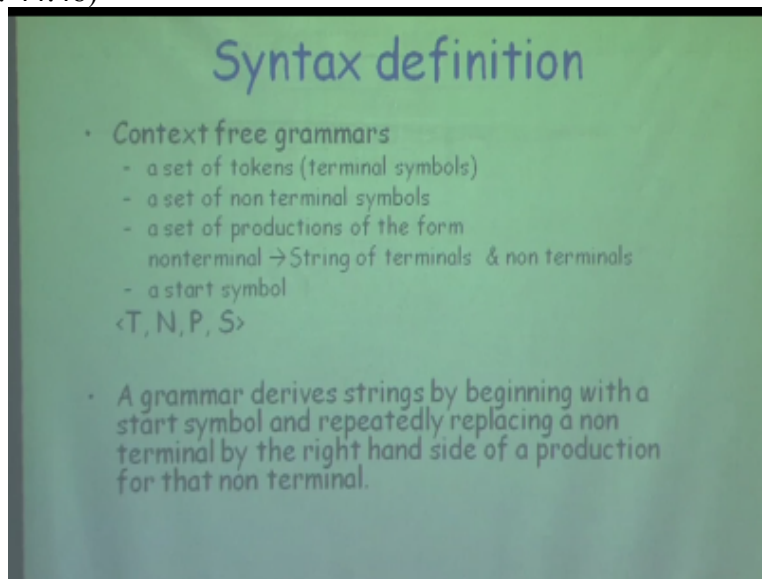
(Refer Slide Time: 42:06)

Limitations of regular languages

- How to describe language syntax precisely and conveniently. Can regular expressions be used?
- Many languages are not regular, for example, string of balanced parentheses
 - $(((((.))))$
 - $\{()^i \mid i \geq 0\}$
 - There is no regular expression for this language
- A finite automata may repeat states, however, it cannot remember the number of times it has been to a particular state

I will go away of counting in a finite state machine or any regular expression so if I want to say do I have a balanced set of brackets that regular expressions cannot pushed out automatically very easy so if I try to capture grammar this is grammar which is many languages are not be regular but this is obviously context streaks so finite automata may repeat states but it cannot remember the number of times it has been to a particular state we have to move to something more powerful framework and more powerful framework is really the next down okay. So you must know very clearly that I can do two things here and I can say what I am doing here cannot be done here and what I am doing here some of those things cannot be done here and they have to be moved to the right so this also brings me back to the overall model of compilation the research that each phase is doing a logical activity well defined logical so that when I start debugging my compiler I know that if I catch a kind of error I know that where should I go should I go and give up this pace or two but this is or debug this phase right. This is clean separation of jobs yes so at least in the context of where parts of sixteen if you finally understand the two boundaries what it cannot do and what should not be doing now you may also ask the question take regular diodes are regular numbers proper subset of this of a subset of context free grammars so why do I have lexical analyzer and just lexical analyzer with my parsers why do I have to face again it is very simple to do polarization one picks T gamma in a more powerful machine then I really need okay. So if I can do something with less complexity let me do it there yeah so this really is a less powerful machinery which is available to me organization yeah I do not want to use this for tokenization similarly if I go to even more powerful mechanism and say that these are all subsets capture everything here this phase will become very complex so we are just trying to do logical

activities and finish those activities so syntax definition this is again I am just recapturing which perhaps you already know from your theory of computation.
(Refer Slide Time: 44:48)



Then context accomplished as far as we are concerned we are going to define context free grammar has consist of a set of vocals so I am not going to talk about all the formal definitions of context free grammar and Boop's and so on but in straight away going to implementations and see that why I am going to specify language issues in context becomes that is why we do TOC before so you have a set of programs which are communistically she has kept Oh non terminal symbols and then you have a set of productions and the productions in context with drama out of the form that left hand side is a non terminal and right inside is a string of terminals and non terminals.

And then you have one of the non terminals as stocks you know right this is what your gamma is so you have a for loop you have a set of terminals on surveillance productions and a start simple this is how you tapped into a context grammar and then once I have two dramas then we say that this grammar actually derives strings by so this is really now you can see that right away I am jumping into some kind of implementation so now we see that if there is something which is specified by the this number or is in the language specifically by grammar then this can always be derived by starting to the start symbol.

Now I am not worried at this point was lying about the efficiency and so on so I can say that let us take the start symbol take all the productions which start with start symbol which starts symbol on the left and if I choose one of those brochures and keep on decreasing a non terminal on the right hand side by one of its production then what I get in the set of strings which can be

derived by this particular start symbol and all such things are going to be part of the language which are specified by this clock right.

This is what my definition is and in fact I mean this tells you that how I am going to do the recognition this is something very neatly captures all the may look very nondescript kind of sentences very neatly captured so what we are going to recognize something so what I do is I say take a string and see if I can derive it from this toxic one and if I can derive it from the start symbol then I try to fill that this is a balancing in this language and if I cannot derive it then I say this is in that so strings which can be derived from the stock symbol of the grammar G form the part of this language energy defined by the top right yeah.

(Refer Slide Time: 47:11)

Examples

- String of balanced parentheses
 $S \rightarrow (S)S \mid \epsilon$
- Grammar
list \rightarrow list + digit
| list - digit
| digit
digit \rightarrow 0 | 1 | ... | 9

Consists of the language which is a list of digit separated by + or -.

So let us look at just one example before we break for today and there is an example and here and set of balanced parentheses so this is S derives as a rule and how many rules do I have here two rules here one is saying that I have a practical expression followed by us or S can also derive as silent and my grammar is another parses another grammar and the list digit list must digit form a list digit list minus digit whatever and what is the language captured by this step of specifications you have all the digits concert of it is special.

which has only two operators plus and minus and what are the opponents here only this thing will be there right and digits I can define by this is consist of the language which is a list of digit separated by plus or minus okay this is really a set of balance credits so what we do in the next class we will start from this and see that even this kind of specification so questions will be very well so I say these specification are some import and I try to see that whether this import along

with this language specified by this particular account so let us break here today and then please note down every one note down.

Acknowledgment

Ministry of Human Resources & Development

Prof. Phalguni Gupta

Co-ordinator, NPTEL IIT Kanpur

Satyaki Roy

Co Co-ordinator, NPTEL IIT Kanpur

Camera

Ram Chandra

Dilip Tripathi

Padam Shukla

Manoj Shrivastava

Sanjay Mishtra

Editing

Ashish Singh

Badal Pradhan

Tapobrata Das

Shuubham Rawat

Shikha Gupta

Pradeep Kumar

K.K Mishra

Jai Singh

Sweety Kanaujia

Aradhana Singh

Sweta

Preeti Sachan

Ashutosh Gairola

Dilip Katiyar

Ashutosh Kumar

Light& Sound

Sharwan

Hari Ram

Production Crew

Bhadra Rao

Puneet Kumar Bajpai

Priyanka Singh

Office

Lalty Dutta

Ajay Kanaujia

Shivendra Kumar Tiwari

Saurabh Shukla

Direction

Sanjay Pal

Production Manager

Bharat Lal

an IIT Kanpur Production

@Copyright reserved