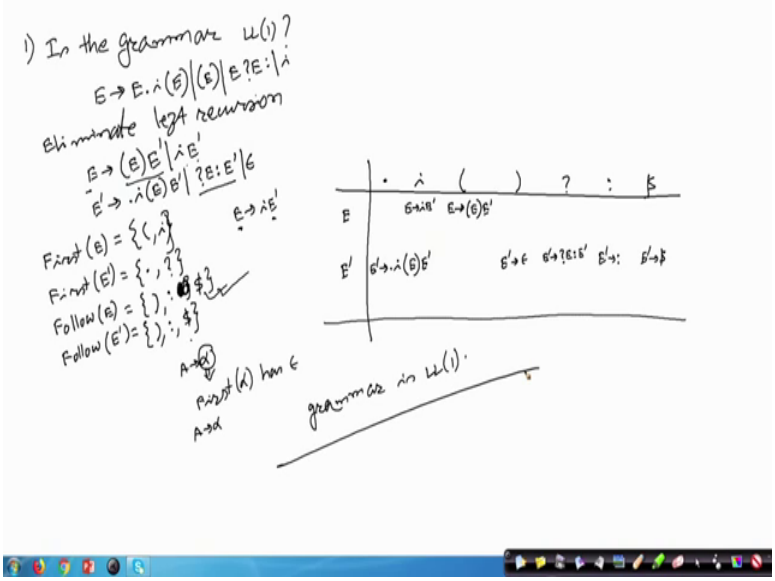
Compiler Design
Prof. Santanu Chattopadhyay
Department of E & EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 33
Parser (Contd.)

So, we have seen our discussions on Parser design. So, in the next few classes so, what we will do is that we will take a few examples and try to solve them, because this particular chapter it requires lot of practicing as I told you. And, you must have also noticed that calculation of this individual parameters, individual sets is often a bit cumbersome. So, I would like that you also do the practices so, that you can be you are quite confident about it. So, to start with, we will solve if a few problems. So, first we solve one problem which is from predictive parsing.

(Refer Slide Time: 00:53)



So, in this problem we will try to see whether a given grammar is whether a given grammar we can have this whether the is LL 1 or not. So, the question is the grammar is LL 1. So, the way to answer this question definitely is to try to construct a LL 1 parsing table and see whether there is any duplicate entry in any of the cells of the table. So, the grammar given is like this E producing E dot i within bracket E where, this dot i then open parenthesis close parenthesis they are all a terminal symbols of the grammar. And, E is the non-terminal symbol with in bracket E or E question mark E colon or i. So, here

this dot this i question mark colon. So, these are all terminal symbols of the grammar. Now, you see that this grammar to check for LL 1 the first obstacle that we have here is that the grammar is left recursive.

So, E producing so, you have you have got this E at the right hand side again. So, first job to do is to eliminate left recursion, the first step for solving this problem is to eliminate left recursion. So, how do you do this? So, there is a set of rules that I have discussed previously. So, if you apply those rules, then it will turn out to be a grammar like this E producing within bracket E and then E dash or i E dash. So, the rules for which immediate left recursion does not appear so, for they are to be taken and this E dash is a new known terminal that has been introduced. And, now E dash can be made to produce dot i within bracket E E dash or question mark E colon then E dash or epsilon.

So, this is the grammar that you get after eliminating the left recursion. So, the next step is to compute the first and follow sets because, we tried to get the predictive parsing table. So, the next step is to get the first and follow set. So, we have got to non-terminals. So, I should have two first sets first of E and first of E dash. So, first of E so, starting with E you see can understand that it will have the symbol open parenthesis and i ok.

So, these are this is the first set, similarly first of E dash, it has got dot and then this question mark. So, this is there in the first of E dash. Now, the follow set follow of E and follow of E dash. So, this is to be computed. Now, follow of E so, if you look into the grammar. So, we have got this close parenthesis because, in this rule it is ending after you get a close parenthesis.

And, also if you look into this rule you see the colon can come after E and this so, whatever. So, then whatever is in first of E dash is follow of E; so, by the because whatever is in. So, if you looking to say this rule so, E producing i E dash. So, whatever region follow of E will be in follow of E dash and then this one follow of E dash you can see that follow of E dash will be equal to follow of E. So, that way so, by this rule say E producing i E dash. So, E producing i E dash.

So, whatever is in follow of E is also in follow of E dash. So, that way follow of E already I have got this so, close parenthesis and colon. So, these two will be there and it has been the start symbol of the grammar. So, dollar will be there in the set and also if so,

this from this rule E producing E within bracket E and then E dash. So, this dollar will also come in the follow set of E.

So, then this if you try to construct the table the parsing table then it will be like this. So, then we have got the symbols dot i then open parenthesis, close parenthesis, then question mark colon and dollar ok. Now, this so, this is by the rules that we have for constructing the predictive parsing table. So, it says that if there is a rule like a producing alpha then and therefore, so you have to look into the first of alpha and for all those first of all those symbols I have to add A producing alpha into the set.

So, if you look into the first rule E producing within brackets E then etcetera then the first. So, this is the alpha part this part is the alpha part. So, the first of it contains the open parenthesis. So, this open parenthesis so, I will add the rule E producing within bracket E then E dash. So, this will be added similarly, when it is applied on the second rule E producing i E dash. So, this will be added here because, I have got i as the star symbol of the i as the i in the first of this i E dash. So, this will be there then for the third rule it says that E dot producing dot i etcetera. So, in the dot I should have this rule added to it.

So, E dash producing dot i within bracket E then E dash this will be there and then this now you see there is a rule like it has producing epsilon. So, there was another rule that if first of alpha if first of alpha contains epsilon. So, if there is a rule such that A producing alpha and first of alpha has got epsilon in it; then I have to add this A producing I have to add this A producing alpha to all the places in the follow of A. So, this E dash producing epsilon so, the in that case alpha becomes equal to epsilon. So, first of alpha definitely contains epsilon. So, this E dash producing epsilon has to be added to all the places where, we have got this where these follow symbols of E dash are coming.

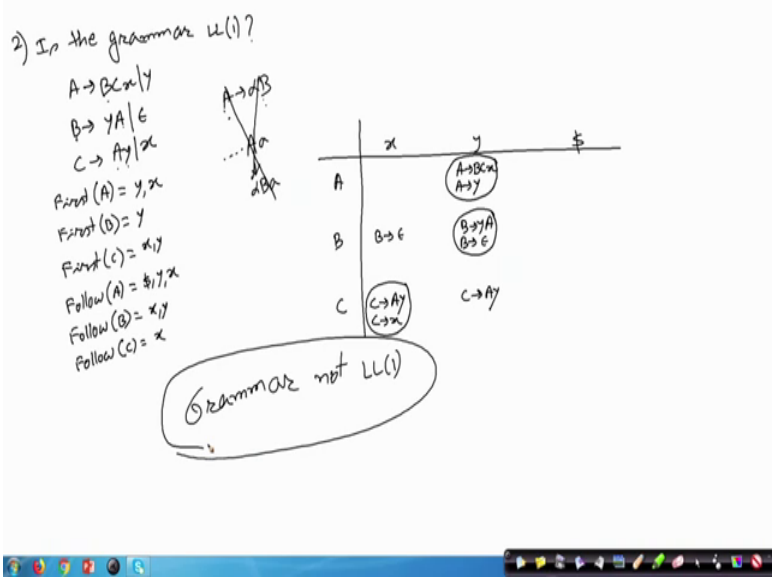
So, follow of E dash contains open close parenthesis, colon and dollar. So, this close parenthesis will have so, I have to have E dash producing epsilon added here, then E dash producing so, for this colon. So, there also I have to have this E dash producing colon appearing there and I have to have these E dash producing epsilon added to this one; E dash producing dollar. Now, once now if you look into this rule E dash producing dot i so, this we have already discussed.

Now, this rule E dash producing question mark then E colon E dash. So, there the first of this are this is the alpha and first of alpha contains question this question mark symbol.

So, this will be added at this point that E dash producing question mark E colon E dash ok. So, this way we can construct this thing this predictive parsing table. So, in fact, in this follow set so, these dollars should not be there because, dollar since the E dash is the star symbol oh sorry sorry sorry; dollar is there in the side because, dollar is the start symbol of the grammar. So, for star symbol of the grammar dollar is there and as a result from this rule E producing i E dash whatever is in follow of E is in follow of E dash so, dollar should also be here so, this was alright.

So, ultimately when we have constructed the parsing table, you see that we have got this in the in this table there is no entry which is multiply define. There is no cell in this particular table where, you have got more than one rules applied rules to be noted there. So, these blank entries are all error. So, that is fine, but there is no duplicate entry. So, this grammar is definitely LL 1 grammar; so, the grammar is LL 1 ok. So, this way you can solve this particular case that this grammar is LL 1. Now, let us look into another grammar and try to see whether that is also LL 1 or not; the grammar is like this that example number 2 question is same.

(Refer Slide Time: 11:41)



That is the grammar is the grammar LL 1 and the grammar given is like this A producing BC x or y B producing y A or epsilon, C producing A y or x ok. So, this grammar we

want to see whether it is LL 1 or not. So, first thing to check whether there is any left recursion. So, there is no left recursion because here I have got A producing BC. So, if I substitute this rule also here it will not lead to any left recursion, similarly here also if I substitute in the place of A B BC. So, it will not start with C. So, this is not a left recursive grammar so, that part is fine. So, we can come to the second step of this problem solving where we try to see what is the first set and follow set.

So, you to have to look into you have to compute this sets first of A first of B first of C, then follow of A follow of B and follow of C. So, these are the sets that we have to construct. Now, first of A is anything that you can start with A. So, it is you have got this y and then you can do it like this that for A can give me y definitely. And, then this if you replace by this BC and then B replaced by epsilon and then C replaced by x.

So, as a result x can also come in the first of A, similarly first of B you have got y ok. So, otherwise a other so, nothing else can come only y can come here and the first of C so, since in starts with A. So, whatever is in first of A is in first of C. So, I have got this symbols x and y both of them. And x from the second rule also this x is automatically coming to the first of C. Now, for the follow set.

Now the follow set since A is the start symbol of the grammar. So, whenever it is not mentioned what is the which one is the start symbol, the first non-terminal appearing on the left hand side or the first rule so, that will be taken as the start symbol. So, dollar is definitely one start symbol dollar, dollar is definitely there in the start symbol A's follow set and then we have to see all the what can follow. Like here if you apply if you look into this rule it says that whatever a so, A is followed by y so, y you will definitely come in the follow of A. Then what is the follow of B?

So, follow of B is whatever is in first of C E can be in follow of B. So, first of C contains x and y. So, follow of B will have x and y in it and follow of C; so, C may be followed by x. So, this is there and from this rule B producing y A whatever is in follow of A will be in follow of B. So, that way so, whatever this B; so, whatever is in follow of whatever is follow of B will be in follow of A.

Like whatever I have got a rule like A producing alpha B, then whatever is in follow of A will be in follow of B. Because, if there is a string where, I have got A followed by some

terminal symbol say a that the next step I can replace it by alpha B a. So, whatever is in follow of A is in can be in follow of B. So, follow of A has got dollar and a.

So, follow of A has got here you see that follow of B whatever is in follow of B can be in follow of A if we go by this rule. So, follow of B has got x and y. So, that will come to follow of A also. So, x y will also be added here. So, this is so, this is the situation. So, I have computed the first and follow sets. Now, after completing the first and follow sets I have to try to construct the LL 1 parsing table.

So, for the LL 1 parsing table construction it go it do it like this. So, I have got the non-terminals A B and C and then I have got the terminal symbols x y and dollar. Now, we come to the first rule A producing BC x. So, I have to see what is the first of BC x. So, first of BC x is equal to first of B first of B is y. So, I will have this rule A producing BC x added here.

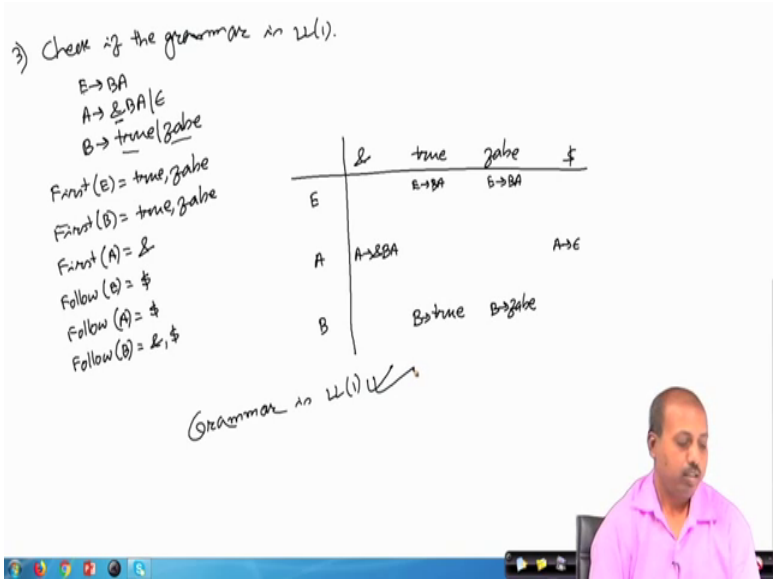
And then so, that will be added here then coming to the second rule A producing y. So, it says that in the first of y this rule should be added so, first of y contains y. So, this rule will be so, this rule will also be added here; A producing y fine then whatever from the second rule we see that B producing y A so, this first of y A contains y. So, B y I will have this rule B producing y A. So, this will be added here now, I have a rule B producing epsilon. So, a B producing epsilon is there so, whatever is in follow of B there this rule has to be added.

So, follow of B contains x and y. So, follow of B will have B producing epsilon. So, this rule will be added here as well as this rule should be added here B producing epsilon, then I will have this third case C producing A y. So, whatever is in first of A y so, first of A y is equal to first of A that is y and x. So, there I will be adding this rule. So, C producing A y and C producing A y so, these two are added and there is a rule C producing x.

So, it says that this rule should be added here also C producing x ok. So, you see that after constructing this table after constructing this table so, we see that several entries are multiply defined. So, this is a multi this there are two rules here, two rules here and two rules here. So, if you make a predictive parser so, parser cannot take a decision like which rule to follow for doing the for parsing the string which rule should be followed. So, naturally so, this is this grammar is not LL 1. So, this grammar is not LL 1. So, this is

the decision that we have for this, that way so, you can always try to do I can try to check whether a grammar is LL 1 or not by constructing the corresponding predictive parsing table. Next we take another example where we try to see whether a given grammar is LL 1 or not.

(Refer Slide Time: 20:05)



And this third example; so, this is the grammar is like this it has got a set of rule. So, again the question is same check if the grammar is LL 1, if the grammar is LL 1. So, the rule is given the production rules are like this E producing BA then A producing ampersand BA or epsilon then B producing true or false ok. So, here this is something like a Boolean grammar and the here all these are non all these are terminal symbols, this ampersand and then true false. So, these are all terminal symbols E B and A they are the non-terminal symbols. Again if you try to check whether this grammar is LL 1 first of all ah; so, the grammar does not have any left recursion.

So, I do not need to do a left recursion removal, but next step is to find the first and follow sets and try to construct the corresponding table. So, if I try to construct so, the first and follow sets. So, first of E then first of B, then first of A then follow of E follow of A and follow of B. So, these are to be computed fine. Now first of E, what is first of E? So, first of E is same as first of B and first of B contains true false.

So, this first of B has got true and false. So, first of E will also have true and false. And what is the first of A? First of A is ampersand so, first of A contains ampersand; so, this is

the thing about this first set. Now, what about the follow set? Follow of E since, is the start symbol of the grammar; so, dollar is definitely there in the follow set.

Now, I have got this E producing BA; that means, whatever is in first of A will be in follow of B. So, first of A has got ampersand so, that will come to follow of B. So, follow of B contains ampersand and then here in this rule E producing BA so, this A can be replaced by epsilon. So, whatever is in follow of E can be in follow of B. So, follow of E contains dollar so, this dollar will come in the follow of B and what about follow of A.

So, follow of A by the same rule whatever is in follow of E will be in follow of A. So, follow of E has got dollar so, this dollar is there in the follow of A ok. So, this way you can construct the first and follow sets for the grammar. So, for the symbols non-terminal symbols; so, you can check that there is no more thing one or more symbol that we can add to the first and follow sets.

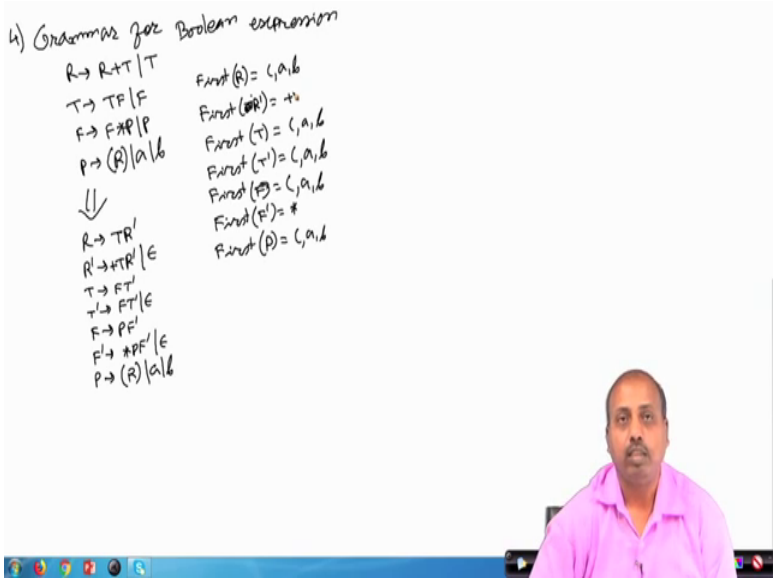
So, next we try to construct the corresponding parsing table to see whether they are duplicate entries ok. So, it is like this so, we have got the symbols non-terminals E A and B and the terminal symbols ampersand, true, false and dollar. Now, this first rule tells me that whatever is in first of B there I have to add E producing BA. So, first of B contains true and false. So, there I should add E producing BA E producing BA so, they are added.

Now, coming to the second rule it says A producing ampersand BA. So, first of ampersand BA contains ampersand so, there I should add this rule that A producing ampersand BA. So, this is added. Now, the second part A producing epsilon. So, since it is epsilon I have to check the follow of A and whatever is in follow of A there I have to add A producing epsilon so, follow of a contains dollar.

So, this A producing epsilon should be added there. So, A producing epsilon is added. Now, coming to the third rule B B producing true or false. So, I will be adding B producing true at this point and B producing false at this point So, all the rules have been taken into consideration so, my table construction is over. So, whatever entries are not defined they are all error entries. So, and you see that there is no duplicate entry. So, this is also this is a grammar is LL 1. So, we can conclude that the grammar is LL 1 grammar

ok. So, next we will be looking into another example of a bigger grammar for which we try to construct the LL 1 parsing table.

(Refer Slide Time: 26:11)



So, this is basically the grammar for Boolean expression, this is the grammar for Boolean expression and we have to construct the corresponding LL 1 parsing table. So, the grammar given is like this R producing R plus T or T, now T producing TF or F then, F producing F star P or P, P producing within bracket R or a or b. This is some sort of Boolean expression so, this is a modified version of that.

So, it is not truly a Boolean expression, but some sort of Boolean expression. So, if we eliminate this left so, first step for getting the um corresponding LL 1 parser is to remove the left recursion. So, if you remove the left recursion so, this grammar can be converted into something like this that R producing T R dash then R dash producing TR dash or epsilon.

Then so, there should be a plus here plus T R dash or epsilon then this T producing F T dash and then T dash producing FT dash or epsilon. Then this F giving PF dash and F dash giving star PF dash or epsilon and then this P giving within bracket R or a or b. So, this is the after eliminating left recursion so, this is the grammar. Now, next part is to compute the first and follow sets. So, the first set R can be computed first of R, first of T first of we go in a order; first of R first of R R dash, then first of T, then first of F dash, then first of F, then first of F dash and first of P ok.

So, first of R so, from the so, first of P is this open parenthesis a b. So, this is open parenthesis a b. So, this is the first of P, then first of P dash F dash is this star from this rule. So, from this one F producing PF dash so, whatever is in first of P is in first of F. So, this one contains open parenthesis a b.

Now, coming to this one whatever is in first of P is in first of T dash. So, first of T dash also contains open parenthesis a b and then from this rule so, so whatever is in first of a so, that is done. So, this is so, whatever is in first of F will be in the first of T. So, again open parenthesis a b. So, they will be in the first of T, then from these rule R dash producing this. So, first of R dash contains plus first of this plus T R dash contains plus. So, this is plus only and from R it says whatever is in first of T will be in first of R. So, first of T contains open parenthesis a b. So, they are included here.

So, we will continue this in the next class.