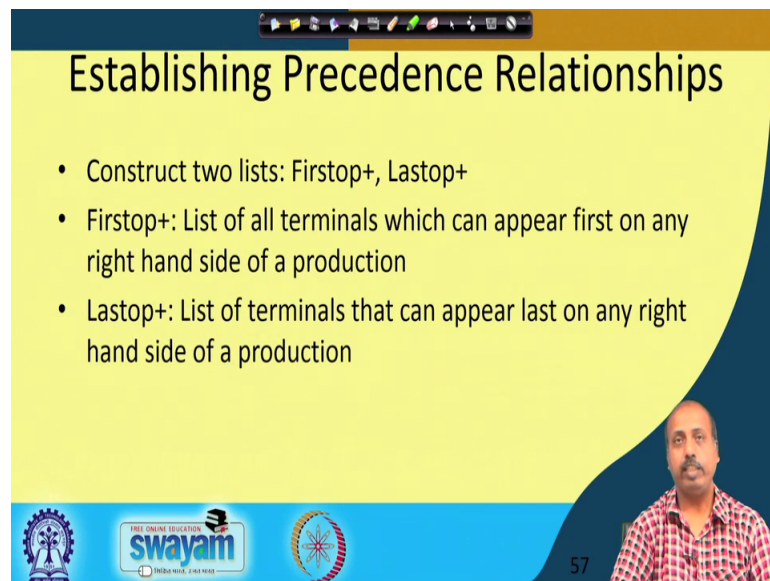


Compiler Design
Prof. Santanu Chattopadhyay
Department of E and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 26
Parser (Contd.)

(Refer Slide Time: 00:15)



Establishing Precedence Relationships

- Construct two lists: Firststop+, Lastop+
- Firststop+: List of all terminals which can appear first on any right hand side of a production
- Lastop+: List of terminals that can appear last on any right hand side of a production

57

So, next we will be seeing how to establish this precedence relationships. So, the previous example that I have we have taken. So, that is from our knowledge in our arithmetic class's ok. So, we know the precedence between the operators and accordingly we could make that table. But in general given a language so, we have to establish precedence relations between the terminal symbols of the grammar. So, how to do that?

So, we first construct two lists; one is known as firststop plus another is known as lastop plus. So, firststop plus is a list of terminals which can appear first on any right hand side of a production. So, you take any if you take any derivation from one particular non terminal symbol.

So, you take any grammar rule so, you so what from the starting with that grammar rule, whatever terminal symbol can appear first on the right hand side so, that is that will be called firststop plus. And similarly lastop so this is a list of terminals that can appear last on any right hand side of a production. So, that will be the lastop plus.

(Refer Slide Time: 01:27)

Firststop and Lastop

- For $X \rightarrow (a...)(Bc)$ put a, B, c in $\text{Firststop}(X)$
- For $Y \rightarrow ...u|...vW$, put u, v, W in $\text{Lastop}(Y)$
- Compute Firststop^+ and Lastop^+ using Closure algorithm
 - Take each nonterminal in turn, in any order and look for it in all the Firststop lists. Add its own first symbol list to any other in which it occurs
 - Similarly process Lastop list
 - Drop all nonterminals from the lists

So, how to construct this lastop plus and firststop plus so, before going to this lastop plus firststop plus so, we have to consider firststop and lastop. So, firststop is like this suppose we have got a grammar rule which is say X producing a etcetera etcetera where a whatever I am writing is in lowercase letter. So, they are terminal symbols and whatever is written in uppercase so, they are non terminal symbol.

So, in this particular rule I have got a as a terminal symbol so, and this B is a non terminal c is a terminal like that. So, what you do you take from the rule the first terminal symbol that can come so, the firststop of X. So, this has got a capital B and c. So, what is the first operator or first symbol that can come first top terminal symbol that can come on the right hand side derived from X ok. So, this is the definition of firststop.

Now, you see if you follow so, from this rule we understand that I can I can get a string where the first character is a first symbol is a. And from this rule it says so, whatever will be there in the firststop of B will be in the firststop of X ok. So, that is why we include b in the firststop of X and later on when we convert in to firststop plus so, those replacements will be done.

Similarly, from the for the lastop so, if there is a rule like this so, Y produce a last symbol is a terminal symbol and here the last symbol is v and W. So, this is this is in that case u v and W will be putting in the lastop of Y.

Now, we compute firststop plus and lastop plus using the closure algorithm. So, closure algorithm is like this that we take a non terminal and then you so, we can take them in any order. So, we order the non terminals in some fashion take the first non terminal and look for it in all the firststop lists. So, in so then we check whether that particular non terminal appears in the firststop list of any other non terminal.

So, if it happens to be so then you add this firststop symbol list to any other in which it occurs. So, like here if you know what is the firststop list of B, then you can add the that firststop list of B to the firststop list of X. So, this way we can compute the firststop plus. Similarly we can create the lastop plus lastop list ok.

So, that will give us lastop plus and then from these two lists so, we drop all the non terminal symbols. Because by the definition of firststop plus and lastop plus so there is no such terminal and non terminal symbols will not be there only terminal. So, we drop all the non terminal symbols from there.

(Refer Slide Time: 04:35)

Example

$E \rightarrow E + T \mid T$	Firststop(E) = {E, +, T}	Lastop(E) = {+, T}
$T \rightarrow T * F \mid F$	Firststop(T) = {T, *, F}	Lastop(T) = {*, F}
$F \rightarrow (E) \mid id$	Firststop(F) = {(, id}	Lastop(F) = {), id}

Firststop+(E) = {E, +, T, *, F, (, id}	Lastop+(E) = {+, T, *, F,), id}
Firststop+(T) = {T, *, F, (, id}	Lastop+(T) = {*, F,), id}
Firststop+(F) = {(, id}	Lastop+(F) = {), id}

Firststop+(E) = {+, *, (, id}	Lastop+(E) = {+, *,), id}
Firststop+(T) = {*, (, id}	Lastop+(T) = {*,), id}
Firststop+(F) = {(, id}	Lastop+(F) = {), id}

60

So, it has take an example and see how this thing happens, say I have got this grammar E producing E plus or T or T, T producing T star for F. F producing within bracket your id. So, firststop of E if you go by this rule it says, E will be there then this plus will be there because this is a first terminal symbol that can come and this is a T, from these rule I can get T. Similarly from T star T producing T star F I will get T and star and from T

producing F I will get F so that is the firstop of T. And firstop of F will be it can be open bracket start or id. So, open parenthesis and id so, these are the firstop of F.

Similarly, if you try to see lastop of E so, it can a lastop can be E plus or this T. So, this plus and T are in the lastop of E; lastop of T is star and F and lastop of F is close parenthesis and id. Then we have to consider the you have to construct the firstop plus so, firstop plus of E is it says that since T appear. So, whatever is appearing in the firstop list of T so, that should appear in the firstop list of E. So, apart from this E plus T so we add this we add this T star and F on to this.

And then in the next round so, since F appears so whatever is in the firstop list of F so, that will appear in here. So, as a result this open bracket open parenthesis and id say they also appear in the firstop plus list of E. Similarly we can construct the firstop plus of T so, which is T star F open parenthesis and id and firstop plus of F which is open parenthesis an id. Then similarly we can construct the lastop plus list. So, lastop plus of E will be whatever is there plus and T, then whatever is there in the lastop of T so, they will appear like star F and from the F I will get this close parenthesis an id so all of them will come.

So, this way we can go on constructing the firstop plus and lastop plus for individual non terminals. And then from this list so, you have to drop the non terminal symbol. So, this E T F are dropped so, you get it like this. Similarly from this you get star open parenthesis an id here you get open parenthesis and id. So, so here also you get the corresponding lastop plus. So, once you have constructed this firstop plus and lastop plus for each of the non terminal symbols. So, we can find out some precedence rule that can that we can be the that can be used for getting an operator precedence parser for the language.


(Refer Slide Time: 07:29)

Constructing Precedence Matrix

- Whenever terminal a immediately precedes nonterminal B in any production, put $a < \alpha$ where α is any terminal in the First $^+$ list of B
- Whenever terminal b immediately follows nonterminal C in any production, put $\beta > b$ where β is any terminal in the Last $^+$ list of C
- Whenever a sequence aBc or ac occurs in any production, put $a \doteq c$
- Add relations $\$ < a$ and $a > \$$ for all terminals in the First $^+$ and Last $^+$ lists, respectively of S

Handwritten notes on the slide:

- $A \rightarrow a B$ with a dashed arrow from a to B and a label $p \rightarrow b$ below it.
- $A \rightarrow a B \alpha$ with a label $a < \alpha$ below it.



So, so this is the rule to construct the precedence matrix. So, it says that whenever a terminal a immediately precedes non terminal B in any production. So, if you have got something like this. So, A producing say $a B$ so, here this terminal a sorry. So, this A producing say $A B X$ then what happens is that this a or say instead of X so, I can write like α so, any symbol any string of so, α is any string of γ symbols.

Now, this B is a non terminal and A is a terminal. So, A is immediately preceding non terminal B in a production then whatever you have in first $^+$ plus of B . So, whatever you have got in first $^+$ plus of B so, so let us make it some other symbol so grammar. So, first $^+$ plus of list suppose it contains the symbol α , then I have to make $a < \alpha$ ok.

Then it says that the second rule tells that if a terminal B immediately follows non terminal C . So, if you have got a rule like this a producing something like say C then b something like this. Then you look into the last $^+$ plus list of C and wherever you have got any β belonging to the last $^+$ plus b last $^+$ plus of C , make them to be higher precedence than b . So, by this rule I have to make this last $^+$ plus and first $^+$ plus these two things.

And if there is a sequence like this so, $a B c$ or ac where B is a non terminal in that case we make the for any production then we make them a and c to be of equal precedence so, this is a third rule. And the fourth rule says that dollar the end of string character. So, that

will that will have lower precedence than all terminals in the firststop plus and lastop plus list of the start symbol of the grammar. Similarly it should be greater than dollar for all the terminals in the firststop plus and lastop plus list of S.

So, it takes the start symbol of the grammar from there we get it. So, using these rules so we can formulate the precedence matrix that can be used by the parser. So, let us see for the example that you are considering how they are going to occur.

(Refer Slide Time: 10:21)

Example (Contd.)

$E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid id$

$Firstop+(E) = \{+, *, (, id\}$
 $Firstop+(T) = \{*, (, id\}$
 $Firstop+(F) = \{(, id\}$

$Lastop+(E) = \{+, *,), id\}$
 $Lastop+(T) = \{*,), id\}$
 $Lastop+(F) = \{), id\}$

	\$	()	id	+	*
\$		<		<	<	<
(<	=	<	<	<
)	>		>		>	>
id	>		>		>	>
+	>	<	>	<		<
*	>	<	>	<	>	

See this is the firstop of plus list for E T F this is the lastop plus list for E T F. Now, let us consider this rule like E producing E plus T F E plus say E producing E plus T. Then the first rule that we had it says that if you have a rule where a terminal a precedes non terminal B, immediately precedes non terminal B. So, that way what qualifies is this plus if we considered the first rule. So, you have to see this plus and T. So, it says that whatever you have got in the firststop plus of T the rule set that whatever you have in the firststop plus of T.

So, this is a should be of lower precedence then that. So, you see here so plus is made to be lower precedence then this plus is to be made lower precedence, then this star open parenthesis an id. So, plus is made lower precedence then open parenthesis id and star fine. And similarly if you come to this rule T producing T star F, then star should be of lower precedence then the firststop plus of F. So, star is of lower precedence then open parenthesis and this id.

Then coming to this rule so, this open parenthesis an E so open parenthesis should be of lower precedence, then all the of all the symbols in the firstop plus of E so, open parenthesis is of lower precedence than this one so this open parenthesis so, plus star open parenthesis an id. So, plus star open parenthesis an id so it is it is less than all of them. So, that finishes complete applying the firstop rule, now with the lastop rules the lastop rules.

So, what will have? So, it says that if you have got a production like this so E so this production so E producing E plus. So, this E and this plus when you consider these two, then the rules said like this that so, this if you take care the lastop plus symbol beta. So, beta should be of higher precedence than b. So, beta should be greater than b. So, here this E so, so plus star bracket close an id, they should be of higher precedence than plus.

So, plus is a higher precedence than plus is of higher precedence than plus sorry so, plus is a higher precedence than plus is of higher precedence than star sorry, star is of higher precedence than plus. Because this is actually the set from where I am taking beta and my rule is beta should be greater than b. So, beta is greater than plus here. Similarly star is greater than plus here, then we have got close parenthesis.

So, close parenthesis is higher than plus and id is also greater than plus so, all the greater, then so I have taken care of. So, this way we can do all these things so for similarly lastop plus of T. So, if from the second rule I will get this T and star ok. Getting this T and star T and star, now lastop plus of this thing has got this star close parenthesis an id.

(Refer Slide Time: 14:25)

Example (Contd.)

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid id$$

$$\text{Firstop}+(E) = \{+, *, (, id\}$$

$$\text{Firstop}+(T) = \{*, (, id\}$$

$$\text{Firstop}+(F) = \{(, id\}$$

$$\text{Lastop}+(E) = \{+, *,), id\}$$

$$\text{Lastop}+(T) = \{*,), id\}$$

$$\text{Lastop}+(F) = \{), id\}$$

$\beta > *$

	\$	()	id	+	*
\$		<		<	<	<
(=	<	<	<
)	>	>		>	>	>
id	>		>		>	>
+	>	<	>	<		<
*	>	<	>	<	>	

Handwritten notes on slide:
 αB^c

So, this so I should if this is said beta from where I am taking beta. So, beta should be greater than star that should be the thing. So, star should be greater than star so, that is done here star should be greater than close parenthesis star should be greater than close parenthesis done here then star should be greater than id. So, so, id should be greater than star. So, id should be greater than star. So, that way it is getting the thing. So, we can see that we can formulate this firstop plus and lastop plus and from there we can make this table.

Now, this particular rule this particular so, F producing [with/within] within bracket E. So, this is of the form that a B c and then it says that a and c should have equal precedence. So, this open parenthesis and close parenthesis so, that is of equal precedence. But it does not mean that close parenthesis so, this particular entry. So, they should also be equalities that is not that is not true ok. So, it should be like a situation is like this. So, if there is open parenthesis there will be a matching close parenthesis, but there cannot be an expression which is like this ok.

So, this is actually excluding that so the if you get an open parenthesis so, in this region so you can get an expression. But here there is a problem so, there must be some operator in between that can that should come. So, there that so the whatever entries are not defined so, they are all errors.

(Refer Slide Time: 15:59)

Example (Contd.)

$E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid id$

$Firstop+(E) = \{+, *, (, id\}$
 $Firstop+(T) = \{*, (, id\}$
 $Firstop+(F) = \{(, id\}$

$Lastop+(E) = \{+, *,), id\}$
 $Lastop+(T) = \{*,), id\}$
 $Lastop+(F) = \{), id\}$

	\$	()	id	+	*
\$		<	<	<	<	<
(>		<	<	<	<
)	>	>		<	<	<
id	>	>	>		<	<
+	>	>	>	>		<
*	>	>	>	>	>	

So, this is error so, this is also error so all these entries that are undefined so, they are all error entries. So, if you if you come to this error entry; that means, there is some problem with the input string. So, that has to be discarded.

(Refer Slide Time: 16:21)

$S \rightarrow (L)a$
 $L \rightarrow L(S)S$
 $Firstop(S) = \{(, a\}$
 $Firstop(L) = \{(, S\}$
 $Lastop(S) = \{), S\}$
 $Lastop(L) = \{), S\}$
 $Firstop+(S) = \{(, a\}$
 $Firstop+(L) = \{(, a\}$
 $Lastop+(S) = \{), a\}$
 $Lastop+(L) = \{), a\}$

$L < A(L)$
 $Lant(L) >$
 $Lant(S) >$
 $Lant(L) >$

	\$	()	a	?
\$		<	<		
(>		<		
)	>	>			
a	>	>	>		
?	>	>	>	>	

So, so next we will be taking an example another example for which we will be constructing the this operator precedence table and operator precedence, how this operator precedence table will work for parsing will work we will take an example.

Suppose let us take an example grammar which is like this. So, S producing within bracket L or a, then L producing L comma S or S so, this is the grammar that we have.

Now, so, first we have to consider at the first of this S and L first of S and first of L. So, first of S so this has got open parenthesis and a. Similarly first of first of L it has got L I will not write this commas, because that will confuse with the comma that we have there so, I will not be writing this commas.

So, let us delete this commas ok. So, let us delete this commas so, I will be as whenever writing separated by a space; that means, that is the next thing that next character that I am writing. So, this has got L then comma and S from this rule L producing S I will have this thing.

Similarly, I can write the last of S to be equal to last of S to be equal to a and close parenthesis and this last of L can be made equal to comma and S last of L is comma and S. So, from this I have to construct the first plus and last plus. So, first plus of S will be equal to so, if does not have any non terminal. So, this is the list first plus of S and first plus of L so, this has got L comma S. So, since S is there so, this bracket start a they will come and the comma will come.

So, this is the first plus of L. Similarly last of S will be equal to a and close parenthesis and last plus so, this is last plus of S and last plus of L will be equal to comma, then last of S will come that is a and close parenthesis. So, they will come so, this is the first plus and last plus rule. Now, based on that we can make the parsing table the or the precedence table sorry. So, you can make the precedence table which will be like this. So, we write down the terminals dollar open parenthesis, close parenthesis, a and comma.

And this side also you have got dollar open parenthesis, close parenthesis, a and comma. So, if I have to fill up this individual entries here ok, now if you apply the rule first rule so, you have got open parenthesis L. So, this first first one it says open parenthesis L. So, whatever you have in the first of plus of L so a should so open parenthesis should be less than first of L ok.

So, first of a first plus of L has got open parenthesis a and comma so, open parenthesis should be less than open parenthesis sorry. So, open parenthesis should be

less than open parenthesis, then we should get this thing this less than so, firstop L has got comma so, comma should be less than open parenthesis.

If this a should be this open parenthesis should be less than comma. So, open parenthesis is less than comma open parenthesis is less than comma. So, these two are done these two are done because this is L so, open parenthesis the and a. So, open parenthesis should be less than a. So, the other entries this one open parenthesis less than a.

Similarly, we have to get the next one so, this L producing L comma S so, between this comma and S. So, comma comes before S so this comma should be less than firstop of S. So, firstop of S firstop plus S has got open parenthesis and a. So, comma should be less than open parenthesis comma is less than open parenthesis and comma is less than a. Comma should be less than a so this should this way I can do this.

Now, this equality these two open parenthesis and close parenthesis coming in this rule S producing L within bracket L. So, that will make this open parenthesis close parenthesis to be of equal precedence. So, that will be done like that. And the rest of the entries like say we have to have this rule say L close parenthesis. So, this L close parenthesis so this will tell that this lastop of lastop of L, they should be of higher precedence than close.

So, lastop L lastop plus of L has got comma and a. So, comma should be higher precedence than bracket close, comma should be of higher precedence than bracket close and this a should be of higher precedence than bracket close and this bracket close should be of higher precedence than bracket close. And then we have got this rule say comma S, so, from this second rule comma S. So, this for lastop S it should be greater than comma.

So, lastop S lastop plus of S has got a and close parenthesis. So, a should be of higher precedence than comma, a should be of higher precedence than comma. And close parenthesis should have higher precedence than comma. So, this is the other rule that we have. Now this one L comma so, L comma. So, this will tell us that this this comma so lastop of L lastop of L this should be of higher higher precedence than comma.

So, lastop of L has got comma. So, comma should be of higher precedence than comma, comma should be of higher precedence than comma then comma should be of higher precedence than comma, then this lastop of L. So, this has got a, a should be of higher

precedence than comma so, that is already done, then close parenthesis should have higher precedence than comma so, that is also already done.

Now, for the dollar part so it is says that since S is the start symbol of the grammar. So, dollar should be less than the open parenthesis. So, you have to see the start firstop plus of S so, that is open parenthesis and a. So, there I should have this relation. And for the lastop plus of S, I have got a and close parenthesis. So, I should have a to be of higher precedence than dollar and close parenthesis to be of higher precedence than dollar. So, using these rules so, we can formulate this table. And we can use this parsing table now for doing the operator precedence parsing.