Lecture - 16

20-05-20

Eliminating Ambiguity

Some grammars that are ambiguous can be converted into unambiguous grammars.

Methods

① Disambiguity rule ② Using precedence and associativity of operators.

Dis-ambiguity Rule

Consider the grammar

$S \rightarrow iCtS \mid iCtSeS \mid a$

(if, then, if, then, else)

$C \rightarrow b$

C, S, a are statement/s

Let us get the parse tree,

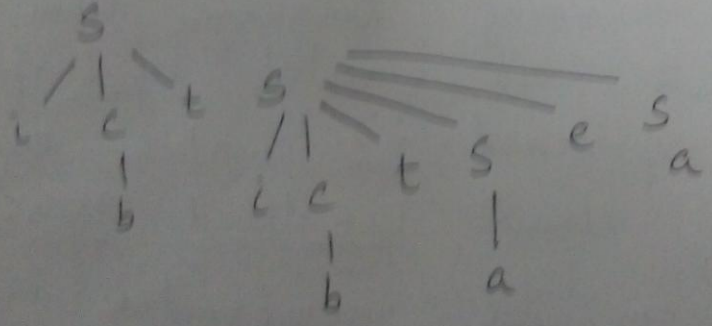

fig 1

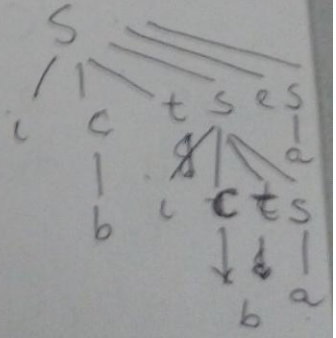

fig 2

As there are two possible parse trees, grammar is ambiguous.

Fig 1 shows that the parse tree associates else with a 2nd if-statement

Fig 2 shows the parse tree associates else with first if-statement

The ambiguity whether to associate else with first if-statement or second if-statement is called "dangling else problem".

↓

how to solve it

↙

by constructing unambiguous grammar

Eliminate ambiguity

$S \rightarrow iCtS \mid iCtSeS \mid a$

$C \rightarrow b$

Sol: In programming languages when if-statements are nested, fig 1 parse tree is prefered. i.e, match each else with closest unmatched then.

This rule can be incorporated into a grammar and ambiguity can be eliminated.

Procedure

Step 1 The matched statement M is an if-else statement where the statement S before else and after else keyword is matched.

M = i C t M e M

Step 2 An unmatched statement U is the one consisting of

(i) simple if-statement where the statement S is matched/unmatched statement. The equivalent production is :

U → i C t S

(ii) if-else statement where the statement before else is matched and statement after else is unmatched. The equivalent production is

U → i C t M e U

Step 3 The matched statement M and unmatched statement U can be obtained using the statement S as S → M/U

So, the final grammar which is un-ambiguous is as under:

$S \rightarrow M \mid U$

$M \rightarrow i\,C\,t\,M\,e\,M$

$U \rightarrow i\,C\,t\,S$

$U \rightarrow i\,C\,t\,M\,e\,U$

Eliminating Ambiguity using Precedence and Associativity

Ex: Convert the given ambiguous grammar to unambiguous grammar.

$E \rightarrow E * E \mid E - E$

$E \rightarrow E \wedge E \mid E / E$

$E \rightarrow E + E$

$E \rightarrow (E) \mid id$

Sol: The grammar can be converted into unambiguous grammar using the precedence of operators as well as associativity.

Step 1 Arrange the operators in increasing order of the precedence along with associativity as shown below:

Operators	Associativity	non-terminal used
+, -	left	E
*, /	Left	T
^	Right	P

Since there are three levels of precedence, we associate three non-terminals: E, T and P. Also an extra non-terminal F, generating basic units in an arithmetic expression.

Step 2: Basic units in expression are id (identifier) and parenthesized expressions. The production corresponding to this is

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

Step 3

The next highest priority operator is ^ and is right associative! so the production must start from the non-terminal P and it should have right recursion

$$P \rightarrow F \hat{} P \mid F$$

Step 4

Next highest priority operators are * and / and they are left associative

So the production must start from the non-terminal T and it should have left recursion.

$$T \rightarrow T*P \mid T/P \mid P$$

Step 5

The next highest priority operators are + and -. And they are left associative. So the production must start from the non-terminal E and it should have left recursion.

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

Step 6

The final grammar which is unambiguous is as under:

$$E \rightarrow E+T \mid E-T \mid T$$
$$T \rightarrow T*P \mid T/P \mid P$$
$$P \rightarrow F \wedge P \mid F$$
$$F \rightarrow (E) \mid id$$