Unambiguous grammar

Martin Kozeny
CSCI 4501: Programming Language Structure
Spring 2011 University of New Orleans

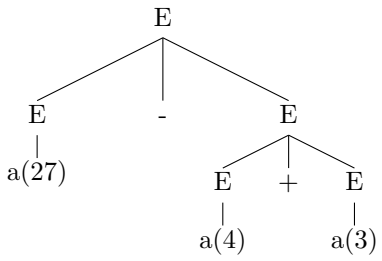
February 21, 2011

1 Ambiguous grammar

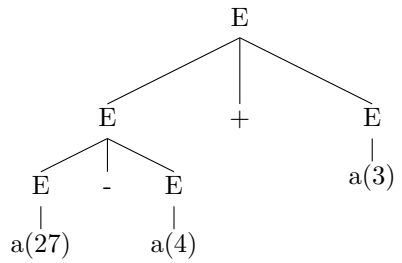
According to regular grammar presented in [1], page 15, we can write similar grammar for arithmetic operations + and −, which will be ambiguous. Context-free grammar $G = (\{E\}, \{a, +, -\}, \{ E \to E + E|E - E|a \}, E)$, where $G = (N, T, R, S)$ is set of N = nonterminals, T = terminals, R = rules and S = start symbol, is ambiguous. Let's take for example expression $27 - 4 + 3$. We can write for this expression more than one derivation tree (trees (a) and (b)). In derivation tree (a) we get the result 20 and in (b) 26. For better readability, we will show rules in rows:

1. $E \to E + E$
2. $E \to E - E$
3. $E \to a$

(a)



(b)

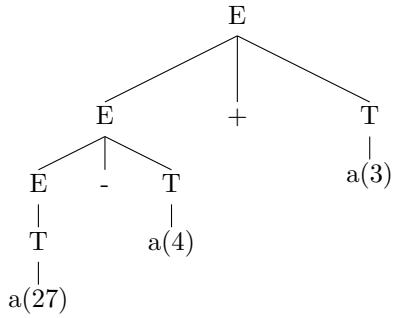


2 Unambiguous grammar

Now we have to change rules in order to get unambiguous grammar. In other words we have to create grammar with left associativity. Due to this fact, we add new nonterminal symbol T (term). This nonterminal symbol guarantees, that on the left side of parse tree will be nonterminal E (expression), which has to be evaluated first in order to get the left operand. Whole grammar will looks that: $G = (\{E, T\}, \{a, +, -\}, \{ E \to E + T|E - T|T, T \to T + a|T - a|a \}, E)$. Using this rules for same expression, we get by derivation tree (c) shown below result 26 because of keeping left associativity. Rules:

1. $E \to E + T$
2. $E \to E - T$
3. $E \to T$
4. $T \to T + a$
5. $T \to T - a$
6. $T \to a$

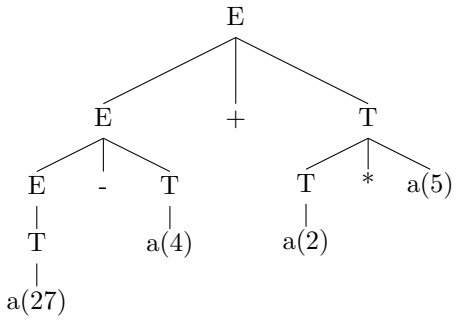
(c)



To add precedence of operations $*$ and / before $+$ and $-$, we have to put this operations into grammar in such way, that operations $*$ and / are evaluated always before $+$ and $-$ and towards operations $+$ and $-$ goes only result of operations $*$ and / keeping left associativity. Let's take e.g. expression $27 - 4 + 2 * 5$. We change grammar to $G = (\{E, T\}, \{a, +, -, *, /\}, \{ E \to E + T|E - T|T, T \to T * a|T/a|a \}, E)$ and for testing expression get right result 33 from derivation tree (d). Rules:

1. $E \to E + T$
2. $E \to E - T$
3. $E \to T$
4. $T \to T * a$
5. $T \to T/a$
6. $T \to a$

(d)



This context-free grammar is not LL1 grammar according to rules for LL1 grammar in [1], page 54. For LL1 grammar hold: right side of rule which is used for expansion of nonterminal symbol at the top of the stack is unambiguously determined on this nonterminal symbol and one ahead readed input symbol (terminal). From shown rules, we can see that this grammar contains right recursion, which has to be for LL1 grammar eliminated.

References

[1] Doc. Ing. Karel Müller, Csc; Czech Technical University in Prague, Faculty of Electrical Engeneering, Department of Computer Science; *Programovací jazyky*, October 2001.