Increasing power of LL(k) parsers

by Jozef Lang(xlangj01), Zoltán Zemko(xzemko01)
Fakulta informačních technológií
Vysoké Učení Technické v Brně
2011

Abstract:

Motivation Non-deterministic top-down parsers are not efficient. They must find first derivation to proceed. Deterministic top-down parsers are faster, because ambiguity is no longer a problem. LL(1) parsers are quite efficient, but there are occasionally situations when LL(1) parsers are not useful and it is better to look ahead *k* symbols with $k > 1$. Here it comes to construction of LL(k) parsers that brings up new problems.

Problem statement For every LL(k) grammar there are LL(k+1) grammars that are not LL(k). With LL(k) grammars the same problems must be solved as with LL(1) grammars; producing parse table is difficult. In LL(1) grammars we can tell whether the grammar is strong-LL(1) and thus LL(1) just by taking first and follow sets. In LL(k) grammars this does not hold, when a grammar is LL(k) it does not imply that it is also a strong-LL(k) grammar. Another disadvantage of LL(k) parser is the size of parse tables. Parse tables of LL(k) grammars are big, but cells are only seldom occupied and thus parsing tables are suitable to table compression.
Second problem discussed; LL(k) grammars provide bounded-look ahead and this is not always sufficient, because discriminating token can be arbitrary far away.
Third problem discussed are extended LL(1) grammars. Several parser generators accept extended context-free grammars and thus these must be transformed into ordinary one without introducing LL(1) conflicts.

Results The problem of the size of LL(k) parsing tables can be avoided by a simple trick; in addition to first, second, third sets etc can be computed. This adjustment transforms the size of the parse table from $O(t^2)$ to O(t) and table is not only smaller, but also easier to generate. This types of grammars are called *Linear-approximate LL(k)* grammars.
The solution to the second discussed problem is to use unbounded-look ahead, but this brings up new problems; they can be solved by approximation CF grammar by regular grammar, where problems brought up by unbounded-look ahead are easily solved. This type of grammars are called *LL-regular* grammars due to approximation by the regular grammars.
Extended parsers work in the following way. Firstly parser transforms extended grammar to an ordinary one and then produces parse table for it. It holds that if resulting context-free grammar is LL(1) then original grammar is ELL(1) (Extended LL(1)).

Conclusion LL(1) parsers are very intuitive, they read input from left to right looking forward only one symbol. But there are situations where LL(k) grammars are not very useful, especially when grammar can contains ε symbol,. The use of LL(k) grammars brings up new problems, some as *LL-regular grammars* are rather the problems in theoretical point of view and in real world are seldom used. Other problems find its application in the real world.

Bibliography:

Grune, D., Jacobs, C.J.H.: Parsing Techniques: A Practical Guide (2nd Edition). Springer, 2008, ISBN 978-0-387-20248-8. Chapter 8.