lex & yacc

Yacc Ambiguities and Conflicts

VYPe presentation abstract

Jan Pačes (xpaces01), Petr Dvořák (xdvora64)

Using yacc to generate a parser based on provided grammar can bring problems with unambiguous rules and conflicts within the grammar. These conflicts can lead to errors of two types: *shift/reduce* and *reduce/reduce* errors. Understanding and handling these conflicts is necessary to generate a valid and reliable parser.

Aim of this presentation is to propose few methods how these ambiguities can be localized and solved and to present common examples of such conflicts. Also how to avoid these conflicts when constructing a grammar.

To actually understand origin of the conflict it is first necessary to understand the model of yacc's operation and how pointers move through the yacc grammar. At the beginning there is only one pointer, however alternatives in the grammar can later create another pointers so there are multiple pointers at the same time. If multiple pointers lead to different reducing rules, a *reduce/reduce* conflict occurs. If one pointer leads to a reduction and another to a shifting rule, we talk about *shift/reduce* conflict.

To choose the next step yacc uses only one following token. This increases the chance of getting into ambiguous situation. Although it might be difficult to localize conflicts directly in the grammar, yacc can provide us with a description of the generated state machine with markers in appropriate states. Localizing *reduce/reduce* conflicts is not so difficult as yacc offers numbers of rules involved in the conflict. Identifying *shift/reduce* conflicts can be bit more difficult as it is necessary to find the reduce rule as well as relevant shift rules and deduce the token stream which caused the conflict.

Probably the three most common situations producing *shift/reduce* conflicts are expression grammars, IF-THEN-ELSE and nested lists of items. Conflicts in expression grammars are mostly *shift/reduce* conflicts that occur due to ambiguity in associativity and which can be easily resolved by specifying left or right precedence. Resolving *shift/reduce* conflicts in IF-THEN-ELSE grammars might be more difficult and might require rewriting the grammar rules or suppressing the conflicts by setting precedence to the token to shift. Finally, resolving *shift/reduce* conflicts in nested lists of items usually requires grammar revision.

The goal of this presenatiton is to show how yacc's operational model and output works, and how to localize, fix and avoid possible ambiguities and conflicts in grammars.