ISSN No. 0976-5697

Volume 6, No. 1, Jan-Feb 2015

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Parsing Techniques and Conflict in LR parsers

Anjali Kedawat
Computer Science and Engineering
Amity University Rajasthan Jaipur,India

Abstract: Basically there are two types of parsers exists:Top down and Bottom Up parsers .Both have their advantages and disadvantages.LR bottom up parser have two type of conflicts :shift/reduce and reduce/reduce.Shift/reduce conflict was removed manually in LALR parser but reduce/reduce conflict still not be solved completely. Automatic generator providing the approach of using first production rule among those which causes reduce/reduce conflict but this approach is not efficient for all grammars.The basic idea to remove this conflict for all types of grammar is to see the last terminal symbol and first terminal symbol of string and according to it use production.

Keywords: conflict, lookahead, parsing, techniques

I. INTRODUCTION

Parsing is the process of determining how a string of terminals can be generated by a grammar

a. Parse tree generation
b. Parsers make a single left-to-right scan over the input tokens, look ahead of one terminal at a time, and construct the parse tree.

A. Parsers:

Top down parser ***Bottom Up parser***

II. PARSING TECHNIQUES

A. Top-down parsers (LL(1), recursive descent):

a. Start at the root of the parse tree and grow toward leaves
b. Pick a production & try to match the input
c. Bad "pick" ! may need to backtrack
d. Some grammars are backtrack-free (predictive parsing)

B. Bottom-up parsers (LR(1), operator precedence)[2]:

a. Start at the leaves and grow toward root
b. As input is consumed, encode possibilities in an internal state
c. Start in a state valid for legal first tokens
d. Bottom-up parsers handle a large class of grammars

III. BOTTOM-UP PARSER

In Bottom-up parsing we start with the sentence and try to apply the production rules in reverse, in order to finish up with the start symbol of the grammar. This corresponds to starting at the leaves of the parse tree, and working back to the root. Bottom-up parsing is also known as **shift-reduce** parsing

A. LR Parser[3]:

LR parsing is a bottom up syntax analysis technique that can be applied to a largeclass of context free grammars. L is for left –to –right scanning of the input and R for constructing rightmost derivation in reverse.

a. Conflicts in SLR Parsers:

a) shift/reduce and reduce/reduce conflicts[4]:

(a). If a state does not know whether it will make a shift operation or reduction for a terminal, we say that there is a **shift/reduce conflict**.
(b). If a state does not know whether it will make a reduction operation using the production rule i or j for a terminal, we say that there is a **reduce/reduce conflict**.
(c). If the SLR parsing table of a grammar G has a conflict, we say that that grammar is not SLR grammar.

Example:[1]
S → L=R
S→ R
L→ *R
L->id
R → .L
I0:S' → .S
S → .L=R
S → .R
L → .*R
L → .id
R → .L
I1: S' → S.
I2: **S → L.=R(causes shift/reduce conflict)**
R->.L
I3: S → R.
I4: L → *.R
R → .L
L→ .*R
L->.id
I_5: L → id
I6: S → L=.R
R → .L
L→ .*R
L->.id
I7: L → *R.
I8: R → L.
I9: S → L=R.

In these conflicts shift/reduce is removed by LALR Parsers but reduce/reduce problem not completely solved manually.

b. Reduce/reduce conflict resolution:

An approach to remove this conflict by seeing the first or last symbol of the i/p string ,rather than seeing first production rule like in Lark[5] and HYACC[6] ,which to be parsed and see which production rule (only those production rule which causes conflict) can generate these symbols. Our approach require less time in comparison to other approaches because in other approaches we use first production rule for removing conflict and if not getting success to parse the whole i/p string we use another production rule (production rule are those which causes reduce/reduce conflict) while in our approach we only need to see the last or first terminal symbol of input string and according to which we use production rule.

IV. REFERENCES

[1] Fraser C., Hanson D., " A Retargetable C Compiler: Design and Implementation",2nd edition,1995 Addison-Wesley Publishing Company.

[2] Bergmann S. "Compiler Design: Theory, Tools, and Examples",4th edition,199, WCB Publishers.

[3] Aho A, Ullman J. , "The Theory of Parsing, Translation, and Compiling.",3rd edition,2009, Prentice-Hall.

[4] A.A Puntambekar, "Compiler Construction",2nd edition,2009 Technical Publication..

[5] Dr. Josef Grosch. "Cocktail A Tool Box for Compiler Construction Lark - An LALR(2) Parser Generator With Backtracking ",CoCoLab – Datenverarbeitung Achern Germany,Document no. 32,2005.

[6] http://hyacc.sourceforge.net, January 27, 2011.