# State Merging in LR Parser under Count based Reduction

R.D. Solomon Raju
Department of Mathematics
Hindustan University

Pawan Kumar
Department of Mathematics
I.I.T. Kharagpur

## ABSTRACT
An LR parser shows error only during scanning input symbol. Error is never shown during the reduction of a handle (substring of stack) into nonterminal. It is because a symbol is put into the stack only when it is guaranteed to be the correct one. If the method of reduction of a handle is known then errors can also be shown during reduction. Hence a wrong symbol can be shifted on the stack and error can be detected during reduce operation. It may permit the merging of few states. The simplest type of reduction scheme is to remove few symbols (the number of symbols equal to the length of the handle) from the top of the stack and push the corresponding nonterminal on the stack. In this paper, a state merging scheme is proposed under this method of reduction.

## General Terms
Your general terms must be any term which can be used for general classification of the submitted material such as Pattern Recognition, Security, Algorithms et. al.

## Keywords
LR Parser, Handle, Stack, CFG (Context Free Grammar)

## 1. INTRODUCTION
LR parsers belong to the class of shift-reduce parsing algorithms [Aho, Denning, and Ullman (1972)]. These are parsers that operate by scanning their input from left-to-right, shifting input symbols onto a pushdown stack until the handle of the current right sentential form is on top of the stack the handle is then reduced. This process is continued either until all of the input has been scanned or the stack contains only the start symbol, or until an error has been encountered. During the 1960s a number of shift-reduce parsing algorithms were found for various subclasses of the context-free grammars. The operator precedence grammars [Floyd[6] (1963]), the simple precedence grammars [Wirth and Weber (1966)], the simple mixed strategy precedence grammars [McKeeman, Horning, and Wortman (1970)], and the uniquely invertible weak precedence grammars [Ichbiah and Morse (1970)] are some of these subclasses. The definitions of these classes of grammars and the associated parsing algorithms are discussed in detail in [Aho and Ullman (1972a)]. In 1965 Knuth defined a class of grammars which he called the LR (k) grammars. These are the context-free grammars that one can naturally parse bottom-up using a deterministic pushdown automaton with k-symbol lookahead to determine shift- reduce parsing actions. This class of grammars includes the other entire shift-reduce parsable grammars and admits of a parsing procedure that appears to be at least as efficient as the shift-reduce parsing algorithms given for these other classes of grammars. [Lalonde, Lee, and Homing[7] (1971)] and [Anderson, Eve, and Homning[4] (1973)] provide some empirical comparisons between LR and precedence parsing that support this conclusion.

## 2. MAIN RESULTS
In this section the new proposed scheme is introduced and rule are given to support with CFG production rules.

The task of LR parser is to show the derivation of a string from a grammar or to show error. If one restricts the aim only on derivation of correct string then two states which do not act differently on same terminal or on nonterminal can always be merged. However if LR parser is constructed by this method of state merging then few of those strings, which can not be generated by the grammar, can also be derived. To stop LR parser from doing so merging is restricted. In construction of LALR parser, two states are merged if they contain similar set of items (items are different only in follow). However this way of merging sometimes introduces conflict which was absent in canonical LR parser e.g. S→aAp S→Bq S→baAq A→bBp A→rs B→ars. Mover over this method fails to capture few merging possibilities. In present research we propose following merging scheme. Moreover

1. Two states should not be merged if their merging leads to non determinism. i.e. Both states show different transition (or reduction) on same symbol (terminal or nonterminal).

2. A State which has item S'→S • $ should not be merged with any other state.

3. A state (P) which contains item $A\rightarrow\alpha\bullet\beta$ should not be merged with any other state (Q) whose distance from a state R is $|\alpha|$ and R contains item $B\rightarrow\gamma\bullet A\delta$. Here A and B are nonterminals and $\alpha$, $\beta$, $\gamma$, $\delta$ denote string of non terminals and terminals. This rule is relaxed in the following three cases.

   (A) On the path from state R to Q, the last transition into Q is by a nonterminal.

   (B) Follow of A in P and R is different.

   (C) State Q has item $A\rightarrow\alpha\bullet\beta$.

4. Two states can always be merged if they act exactly in same way on all symbols. If one state shows error on some symbol then the other state also shows error. If one state performs transition on some symbol then same transition is performed by the other also state. This rule will not permit any state merging on canonical LR parser. However if state merging on canonical LR parser is done (using above rules) then additional merging can be done using this rule.

5. Decision about merging pair of states should not be taken in parallel. Moreover these rules may not be applicable if after merging a pair of states the other pair of states is merged.

## 3. RULES

### 3.1 Rule 1

Let us construct canonical LR parser for the grammar 1 .S→ abcde 2. S→pqd

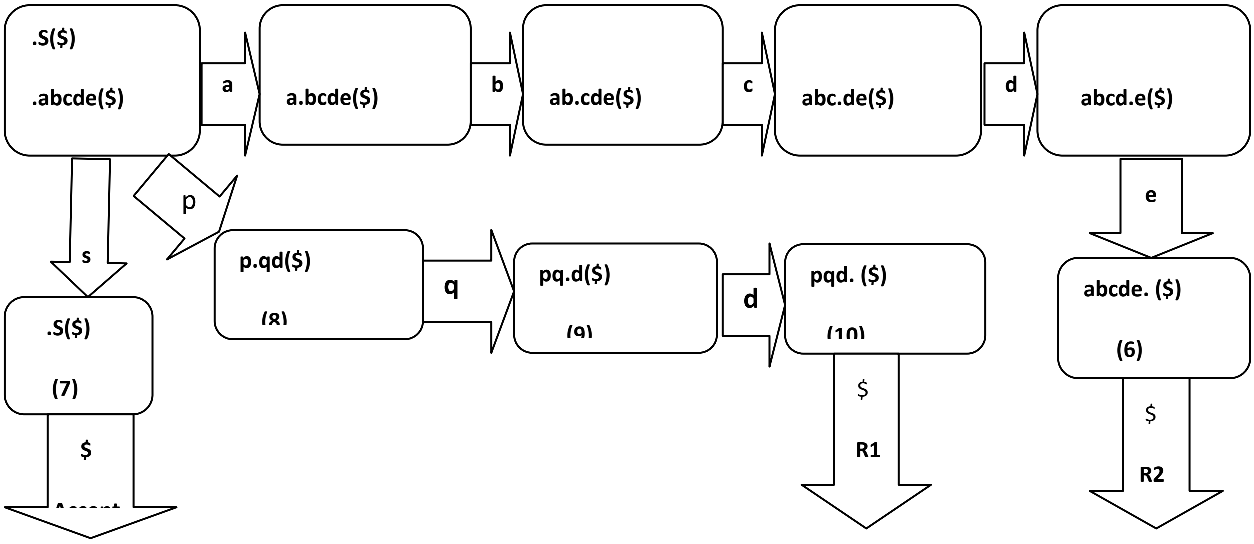


**Fig 3.1**

| STATES | ACTION | | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **e** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3 | | | S4 | | | | | | | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 9 | | | | | | | | S10 | | |
| 10 | | | | | | | | | R2 | |

**Table 3.1**

If we merge states 4 and 9 then in the merged state the action on the terminal 'd' becomes nondeterministic. It is because both states 4 and 9 show transition on 'd'.

| STATES | ACTION | | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **e** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3 | | | S4 | | | | | | | |
| 4 ,9 | | | | S5 | | | | S10 | | |
| 5 | | | | | S6 | | | | | |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | S9 | | | | |
| 10 | | | | | | | | | R2 | |

**Table 3.2**

Merging of states 6 and 10 gives rise to reduce-reduce conflict. It is due to reduction on $ is permitted in both states.

| **STATES** | **ACTION** | | | | | | | | | **GOTO** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **e** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3 | | | S4 | | | | | | | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 ,10 | | | | | | | | | R1,R2 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | S9 | | | | |
| 9 | | | | | | | | S10 | | |

**Table 3.3**

## 3.2 Rule 2

When we merge states 3 and 7 we get the following:

| **STATES** | **ACTION** | | | | | | | | | **GOTO** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **E** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3,7 | | | S4 | | | | | | Acc | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 8 | | | | | | S9 | | | | |
| 9 | | | | | | | | S10 | | |
| 10 | | | | | | | | | R2 | |

**Table 3.4**

The above parser shows that string "ab" is accepted.

| **Stack** | **Input** |
|---|---|
| 1 | ab$ |
| 1a2 | b$ |
| 1a2b3 | $ |
| 1S3 | $ accepted |

Hence the state which has item S'→S•$ should not merged with any other state.

### *3.2.1 Count based reduction*

In the parser for the grammar 1.S→ abcde 2.S→pqd, when the states 3 and 9 then neither rule 1 or 2 is violated.

| **STATES** | **ACTION** | | | | | | | | | **GOTO** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **E** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3,9 | | | S4 | | | | | S10 | | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | S9 | | | | |
| 10 | | | | | | | | | R2 | |

**Table3.5**

However it leads to the acceptance of a wrong string abd. Its parsing is as follows:

| **Stack** | **Input** |
|---|---|
| 1 | abd$ |
| 1a2 | bd$ |
| 1a2b3 | d$ |
| 1a2b3d10 | $ |
| 1S7 | $ accepted |

Here the string abd is reduced by S→pqd. Because |pqd|=|abd|=3. Here 3 symbols are popped during reduction. Similarly string pqcde will be accepted because |abcde|=|pqcde|=5. However if we would have done string compare during reduction error had been detected (Dillip[2]). However comparison of handle with substring on stack would require additional time hence parsing is slowed.
However merging of states 2 and 9 does not create any problem.

| **STATES** | **ACTION** | | | | | | | | | **GOTO** |
|---|---|---|---|---|---|---|---|---|---|---|
| | **a** | **b** | **c** | **d** | **e** | **p** | **q** | **d** | **$** | **S** |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2,9 | | S3 | | | | | | S10 | | |
| 3 | | | S4 | | | | | | | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | S9 | | | | |
| 10 | | | | | | | | | R2 | |

**Table3.6**

Let us see parsing of valid string pqd.

| **Stack** | **Input** |
|---|---|
| 1 | pqd$ |
| 1p8 | qd$ |
| 1p8q2 | d$ |
| 1p8q2d10 | $ |
| 1S7 | $ accepted |

On the other hand an invalid string “ad” is rejected.

| **Stack** | **Input** |
|---|---|
| 1 | ad$ |
| 1a2 | d$ |
| 1a2d10 | $ |

Now attempt is made to perform reduction by the rule S→pqd. Hence |pqd|=3 symbols from the stack are popped. But it is not possible. Hence error is reported. An important thing to note is that error will be shown during reduction. While in original LR parser (before state merging) error will produced when d is seen in state 2 (while shifting). Hence error reporting has been delayed.

Let us see parsing of pqbcde.

| **Stack** | **Input** |
|---|---|
| 1 | pqbcde$ |
| 1p8 | qbcde$ |
| 1p8q9 | bcde$ |
| 1p8q9b3 | cde$ |
| 1p8q9b3c4 | de$ |
| 1p8q9b3c4d5 | e$ |
| 1p8q9b3c4d5e6 | $ |
| 1p8S | error |

In state 6 reduction by S→ abcde is done by popping |abcde|=5 symbols from the stack. However in state 8 one can not go to any state on the arrival of nonterminal S. Hence error is reported. In original LR parser (before state merging) error will produced when b is seen in state 9 (while shifting). In the LR parser after state merging the reduction into S will be successful. The reported error in original parser will be "d is replaced by b". In modified parser error reporting will be misleading.

## 3.3 Rule 3

Following is LR parser for the grammar 1.S→hAgBkm 2.A→abcd 3. B→pq

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **h** | **g** | **k** | **m** | **a** | **B** | **c** | **d** | **p** | **q** | **$** | **S** | **A** | **B** |
| 1 | S2 | | | | | | | | | | | 14 | | |
| 2 | | | | | S8 | | | | | | | | 3 | |
| 3 | | S4 | | | | | | | | | | | | |
| 4 | | | | S5 | | | | | S12 | | | | | 5 |
| 5 | | | S6 | | | | | | | | | | | |
| 6 | | | | S7 | | | | | | | | | | |
| 7 | | | | | | | | | | | R1 | | | |
| 8 | | | | | | S9 | | | | | | | | |
| 9 | | | | | | | S10 | | | | | | | |
| 10 | | | | | | | | S11 | | | | | | |
| 11 | | R2 | | | | | | | | | | | | |
| 12 | | | | | | | | | | S13 | | | | |
| 13 | | | R3 | | | | | | | | | | | |

**Table3.7**

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **h** | **g** | **k** | **m** | **a** | **B** | **C** | **d** | **p** | **q** | **$** | **S** | **A** | **B** |
| 1 | S2 | | | | | | | | | | | 14 | | |
| 2 | | | | | S8 | | | | | | | | 3 | |
| 3 | | S4 | | | | | | | | | | | | |
| 4 | | | | S5 | | | | | S12 | | | | | 5 |
| 5 | | | S6 | | | | | | | | | | | |
| 6 | | | | S7 | | | | | | | | | | |
| 7 | | | | | | | | | | | R1 | | | |
| 8 | | | | | | S9 | | | | | | | | |
| 9 | | | | | | | S10 | | | | | | | |
| 10,12 | | | | | | | | S11 | | S13 | | | | |
| 11 | | R2 | | | | | | | | | | | | |
| 13 | | | R3 | | | | | | | | | | | |

After merging states 10 and 12 it looks as

**Table3.8**

In above LR parser state 2 has item S→h•AgBkm and state 12 is at a distance of |abc|=3 from state 2. Since state 10 has item A→abc•d hence its merger with state 10 leads to acceptance of invalid string habcdgpdgpqkm.

| **Stack** | **Input** |
|---|---|
| 1h2a8b9c10d11 | gpdgpqkm$ |
| 1h2A3 | gpdgpqkm$ |
| 1h2A3g4 | pdgpqkm$ |
| 1h2A3g4p10 | dgpqkm$ |
| 1h2A3g4p10d11 | gpqkm$ (Agpd is reduced to A) |
| 1h2A3 | gpqkm$ |
| 1h2A3g4 | pqkm$ |
| 1h2A3g4p10 | qkm$ |
| 1h2A3g4p10q13 | km$ |
| 1h2A3g4B5 | km$ |
| 1h2A3g4B5k6 | m$ |
| 1h2A3g4B5k6m7 | $ |
| 1S14 | $ accepted |

Let us construct LR parser for the grammar1.S→pqAghij 2.A→abcd

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **q** | **g** | **h** | **i** | **j** | **A** | **b** | **c** | **d** | **$** | **S** | **A** |
| 1 | S2 | | | | | | | | | | | 9 | |
| 2 | | S3 | | | | | | | | | | | |
| 3 | | | | | | | S10 | | | | | | 4 |
| 4 | | | S5 | | | | | | | | | | |
| 5 | | | | S6 | | | | | | | | | |
| 6 | | | | | S7 | | | | | | | | |
| 7 | | | | | | S8 | | | | | | | |
| 8 | | | | | | | | | | | R1 | | |
| 9 | | | | | | | | | | | Acc | | |
| 10 | | | | | | | | S11 | | | | | |
| 11 | | | | | | | | | S12 | | | | |
| 12 | | | | | | | | | | S13 | | | |
| 13 | | | R2 | | | | | | | | | | |

**Table 3.9**

After merging states 6 and 12 the parser accepts a wrong string pqabcdghdghij. It is because state 12 has item A→abc•d and state 6 has item S→pqAgh•ij. The state 6 is at the distance |abc|=3 from state 3, which has item S→pq•Aghij.

Let us construct LR parser for the grammar 1.S→pqAghij 2.A→abcd

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **q** | **g** | **h** | **i** | **j** | **A** | **b** | **c** | **d** | **$** | **S** | **A** |
| 1 | S2 | | | | | | | | | | | 9 | |
| 2 | | S3 | | | | | | | | | | | |
| 3 | | | | | | | S10 | | | | | | 4 |
| 4 | | | S5 | | | | | | | | | | |
| 5 | | | | S6 | | | | | | | | | |
| 6 | | | | | S7 | | | | | | | | |
| 7 | | | | | | S8 | | | | | | | |
| 8 | | | | | | | | | | | R1 | | |
| 9 | | | | | | | | | | | Acc | | |
| 10 | | | | | | | | S11 | | | | | |
| 11 | | | | | | | | | S12 | | | | |
| 12 | | | | | | | | | | S13 | | | |
| 13 | | | R2 | | | | | | | | | | |

**Table 3.9**

After merging states 6 and 12 the parser accepts a wrong string pqabcdghdghij. It is because state 12 has item A→abc•d and state 6 has item S→pqAgh•ij. The state 6 is at the distance |abc|=3 from state 3, which has item S→pq•Aghij.

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **q** | **g** | **h** | **i** | **j** | **A** | **b** | **c** | **d** | **$** | **S** | **A** |
| 1 | S2 | | | | | | | | | | | 9 | |
| 2 | | S3 | | | | | | | | | | | |
| 3 | | | | | | | S10 | | | | | | 4 |
| 4 | | | S5 | | | | | | | | | | |
| 5 | | | | S6 | | | | | | | | | |
| 6,12 | | | | | S7 | | | | | S13 | | | |
| 7 | | | | | | S8 | | | | | | | |
| 8 | | | | | | | | | | | R1 | | |
| 9 | | | | | | | | | | | Acc | | |
| 10 | | | | | | | | S11 | | | | | |
| 11 | | | | | | | | | S12 | | | | |
| 13 | | | R2 | | | | | | | | | | |

**Table3.10**

A potentially wrong string is pqabcdghcdghij. But it is rejected.

| **Stack** | **Input** |
|---|---|
| 1 | pqabcdghcdghij$ |
| 1p2q3a10b6c12d13 | ghcdghij$ |
| 1p2q3A4 | hcdghij$ |
| 1p2q3A4g5 | cdghij$ |
| 1p2q3A4g5h6 | dghij$ |
| 1p2q3A4g5h6c12 | ghij$ |
| 1p2q3A4g5h6c12d13 | ghij$ |
| 1p2q3A4A | error |

Since in state 4 one can not go to any state on nonterminal A.

### 3.3.1 Rule 3 – Relaxation (A)

Let a grammar be 1.S → pqAgHij 2.A → abcd 3.H → h. It is similar to the previous grammar.

| **STATES** | **ACTION** | | | | | | | | | | **GOTO** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **q** | **g** | **i** | **j** | **a** | **b** | **c** | **d** | **h** | **$** | **S** | **A** | **H** |
| 1 | S2 | | | | | | | | | | | 9 | | |
| 2 | | S3 | | | | | | | | | | | | |
| 3 | | | | | | S10 | | | | | | | 4 | |
| 4 | | | S5 | | | | | | | | | | | |
| 5 | | | | | | | | | | S14 | | | | |
| 6 | | | | S7 | | | | | | | | | | |
| 7 | | | | | S8 | | | | | | | | | |
| 8 | | | | | | | | | | | R1 | | | |
| 9 | | | | | | | | | | | Acc | | | |
| 10 | | | | | | | S11 | | | | | | | |
| 11 | | | | | | | | S12 | | | | | | |
| 12 | | | | | | | | | S13 | | | | | |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | | | R2 | | | | | | | | | | | |
| 14 | | | | R3 | | | | | | | | | | |

**Table 3.11**

Following is LR parser after merging states 6 and 12.

| **STATES** | **ACTION** | | | | | | | | | | | **GOTO** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **p** | **q** | **g** | **i** | **j** | **a** | **b** | **c** | **d** | **h** | **$** | **S** | **A** | **H** |
| 1 | S2 | | | | | | | | | | | 9 | | |
| 2 | | S3 | | | | | | | | | | | | |
| 3 | | | | | | S10 | | | | | | | 4 | |
| 4 | | | S5 | | | | | | | | | | | |
| 5 | | | | | | | | | | S14 | | | | |
| 6,12 | | | | S7 | | | | | S13 | | | | | |
| 7 | | | | | S8 | | | | | | | | | |
| 8 | | | | | | | | | | | R1 | | | |
| 9 | | | | | | | | | | | Acc | | | |
| 10 | | | | | | | S11 | | | | | | | |
| 11 | | | | | | | | S12 | | | | | | |
| 13 | | | R2 | | | | | | | | | | | |
| 14 | | | | R3 | | | | | | | | | | |

**Table3.12**

Let us see parsing of the string pqabcdghdghij.

| **Stack** | **Input** |
|---|---|
| 1p2q3A4 | ghdghij$ |
| 1p2q3A4g5 | hdghij$ |
| 1p2q3A4g5h14 | dghij$ |

Now error is reported. It is because in state 14 no action can be taken on the arrival of input symbol 'd'. It is to be noted the same string was parsed when the grammar was S → pqAghij A → abcd and same state merging was done. This example shows relaxation of 3(A).

### *3.3.2 Rule 3 - Relaxations (B)*

1.S→tabcde 2.S→tApqrs 3.S→tghijk 4.S→bApt 5.A→abcd

| States | Action | | | | | | | | | | | | | | | Goto | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | t | a | b | c | d | e | p | q | r | s | g | h | i | j | k | $ | S | A |
| 1 | S2 | | | | | | | | | | | | | | | | 28 | |
| 2 | | | | | | | | | | | | | | | | | | 3 |
| 3 | | | | | | | S4 | | | | | | | | | | | |
| 4 | | | | | | | | S5 | | | | | | | | | | |
| 5 | | | | | | | | | S6 | | | | | | | | | |
| 6 | | | | | | | | | | S7 | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | R2 | | |
| 8 | | | S9 | | | | | | | | | | | | | | | |
| 9 | | | | S10 | | | | | | | | | | | | | | |
| 10 | | | | | S11 | | | | | | | | | | | | | |
| 11 | | | | | | S12 | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | R1 | | |
| 13 | | | | | | | | | | | | S14 | | | | | | |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | | | | | | | | | | | | | S15 | | | | | |
| 15 | | | | | S23 | | | | | | | | | S16 | | | | |
| 16 | | | | | | | | | | | | | | | S17 | | | |
| 17 | | | | | | | | | | | | | | | | R3 | | |
| 20 | | S21 | | | | | | | | | | | | | | | | |
| 21 | | | | S22 | | | | | | | | | | | | | | |
| 23 | | | | | | | R5 | | | | | | | | | | | |
| 24 | | S20 | | | | | | | | | | | | | | | | 25 |
| 25 | | | | | | | S26 | | | | | | | | | | | |
| 26 | | S27 | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | R4 | | |
| 28 | | | | | | | | | | | | | | | | acc | | |

**Table3.13**

When state 15 and 22 are merged then invalid string tghidpqrs is accepted. It is because state 2 has item S→t•Apqrs (and S→t•ghijk) and state 15 has item S→tghi•jk. The state 22 has item A→abc•d. The path length from state 2 to 15 is |abc|=3

| **Stack** | **Input** |
|---|---|
| 1t2g13h14i15d23 | pqrs$ |
| 1t2A3 | pqrs$ (ghid reduces to A) |
| 1t2A3p4q5r6s7 | $ accepted |
| 1S28 | |

However when S→bApt is replaced by S→bAqt then the same string is not accepted. It is because fellow of A is state 2 and 22 become different (p and q). It is rule 3 relaxation (A).

The LR parser is same except in state 23 reduce action is done on 'q' (in place of 'p')

| **Stack** | **Input** |
|---|---|
| 1t2g13h14i15d23 | pqrs$ reduce action can not be taken |

### 3.3.3 Rule 3 - Relaxation (C)

Following is canonical LR parser for the grammar 1.S→gApt 2.S→gabc 3.S→Aq 4.A→ab

| **States** | **Action** | | | | | | | | **Goto** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **g** | **p** | **t** | **g** | **a** | **b** | **c** | **q** | **$** | **S** | **A** |
| 1 | | | | | S9 | | | | | 11 | 12 |
| 2 | | | | | S6 | | | | | | 3 |
| 3 | | S4 | | | | | | | | | |
| 4 | | | S5 | | | | | | | | |
| 5 | | | | | | | | | R1 | | |
| 6 | | | | | | S7 | | | | | |
| 7 | | R4 | | | | | S8 | | | | |
| 8 | | | | | | | | | R2 | | |
| 9 | | | | | | S10 | | | | | |
| 10 | | | | | | | | R4 | | | |
| 11 | | | | | | | | | Acc | | |
| 12 | | | | | | | | S13 | | | |
| 13 | | | | | | | | | R3 | | |

**Table3.14**

Here state 7 and 13 can be merged using rule3-Relaxation (c).After that 6 and 12 can also be merged (using rule 4). The merging of state 6 and 12 in a original parser will cause non determinism.

| **States** | **Action** | | | | | | | | **Goto** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **g** | **p** | **t** | **a** | **b** | **c** | **h** | **q** | **$** | **S** | **A** |
| 1 | | | | | S9 | | | | | 11 | 12 |
| 2 | | | | | S6 | | | | | | 3 |
| 3 | | S4 | | | | | | | | | |
| 4 | | | S5 | | | | | | | | |
| 5 | | | | | | | | | R1 | | |
| 6,12 | | | | | | S7 | | S13 | | | |
| 7,13 | | R4 | | | | | S8 | | R3 | | |
| 8 | | | | | | | | | R2 | | |
| 9 | | | | | | S10 | | | | | |
| 10 | | | | | | | | R4 | | | |
| 11 | | | | | | | | | Acc | | |

**Table3.15**

## 3.4 Rule 4

Let in the LR parser for grammar 1.S→ abcde 2.S→pqd states 5 and 10 are merged.

| STATES | ACTION | | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | p | q | d | $ | S |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3 | | | S4 | | | | | | | |
| 4 | | | | S5 | | | | | | |
| 5,10 | | | | | S6 | | | | R2 | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 9 | | | | | | | | S10 | | |

**Table4.1**

Now the states 4 and 9 can also be merged using rule 4

| STATES | ACTION | | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | p | q | d | $ | S |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3 | | | S4 | | | | | | | |
| 4,9 | | | | S5 | | | | S10 | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 10 | | | | | | | | | R2 | |

**Table4.2**

It is to be noted that before merging 5 and 10 the merging of 4 and 9 will cause non determinism.

## 3.5 Rule 5

In the LR parser for the grammar 1.S→ abcde 2.S→pqd merging of states 2 and 9 does not cause any problem because potentially wrong strings ad and pqbcde will be rejected during count based reduction.

| STATES | ACTION | | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | p | q | d | $ | S |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2,9 | | S3 | | | | | | S10 | | |
| 3 | | | S4 | | | | | | | |
| 4 | | | | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 10 | | | | | | | | | R2 | |

**Table5.1**

Similarly merging of states 3 and 4 does not cause any problem because potentially wrong strings abde, abccde, abcccde, will be rejected.

| STATES | ACTION | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | p | q | d | $ | S |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | | | |
| 3,4 | | | S4 | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 9 | | | | | | | | S10 | | |
| 10 | | | | | | | | | R2 | |

**Table5.2**

However merging both pair of states will cause problem. Here string pqbde is accepted. If state 2 and 9 are merged and then merging of states 3 and 4 is prohibited because P (state 4) has item S→abc•de and Q (state 3) is at a distance of |abc|=3 form R (state 1) which has item S'→•S. The distance is 3 via path 1p8q2b3.

If first state 3 and 4 are merged then we do not have any choice of P, Q and R to prohibit merging of 2 and 9. Hence given set of rules may not remain applicable after merging a pair of states.

| STATES | ACTION | | | | | | | | GOTO |
|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | d | e | p | q | d | $ | S |
| 1 | S2 | | | | | S8 | | | | 7 |
| 2 | | S3 | | | | | | S10 | | |
| 3 | | | S4 | S5 | | | | | | |
| 5 | | | | | S6 | | | | | |
| 6 | | | | | | | | | R1 | |
| 7 | | | | | | | | | Acc | |
| 8 | | | | | | | S9 | | | |
| 10 | | | | | | | | | R2 | |

**Table5.3**

## 4. DISCUSSION

Let a state P has item A→α•β(F) and another state R has item B→γ•Aδ. Suppose stack contains states R and Q and some sting φ is in between them. If states P and Q are merged then (wrong) string β can be pushed on the stack. If the next input symbol t∈F and |φ| = |α| then by count based reduction φβ is reduced to A. Now top state on the stack is R, which has item B→γ•Aδ hence transition to a state (T) with item B→γA•δ is made. If t∈First(δ) then no error is shown.

(i) If t∉First(δ) then transition from state T can not be made. Hence error is shown. It is relaxation 3(B). Here fellow of A in P and R are different.

(ii) Suppose on the path from R to Q transition into Q is by a nonterminal (K) and if follow(K) in state Q and first (β) are

(a) Disjoint: then error is shown during reduction of K.

(b) Not disjoint: then merging will be prohibited by rule 1.

In any case error is shown. It is relaxation 3(A).

(iii) If Q has item A→α•β then φ=α hence reduction of φβ into A is not an error. It is relaxation 3(C)

In the grammar S→abcde S→pqr. Let state P has item S→pq•r($) and R has item S'→•S($) (so S→•abcde also). The state Q, which has item S→ab•cde is at a distance of 2=|pq|, from R. Hence the merging leads to acceptance of wrong strings abr.

In grammar S→pqAghij A→abcd the state P has item A→abc•d and state R has S→pq•Aghij. The state Q which has item S→pqAgh•ij is at a distance of 3=|abc|. Hence merging of states P and Q can make stack contents pqAghd. The count based reduction scheme will make stack contents pqA.

In grammar S→pqAgHij A→abcd H→h stack pqAgh will be modified to pqAgH only when next input is i. Hence stack can not become pqAghd. It is relaxtion 3(A).

In grammar S→hAgBkm A→abcd B→pq the state R has item S→h•AgBkm. Its distance from the state, which has item S→hAg•Bkm (and B→•pq) is 2. Its distance from from state

Q (which has item B→p•q) is 1. Hence path length form R to Q is 3. When stack has hAgp and if states with items B→p•q and A→abc•d are merged hence next stack configuration will be hAgpd. It can be modified to hA.

In grammar S→tabcde S→tApqrs S→tghijk S→bApt A→abcd the state R has item S→t•Apqrs. R also has item S→t•ghijk. Hence from R the distance of state Q which has item S→tghi•jk is 3. Its merger with state P, which has item A→abc•d(p) creates problem. It is because when tghid is on stack and remaining input string is pqrs then stack is modified as tApqrs. However when S→bApt is replaced by S→bAqt then state P has item A→abc•d(q). Hence when tghid is on stack and remaining input string is pqrs then stack is not modified as tA and error is shown. It is relaxation 3(B).

In grammar S→gApt S→gabc S→Aq A→ab. The state P has item A→ab•(q) and state R has item S→g•Apt (and S→g•abc). The state Q which has item S→gab•c (and A→ab•(p)) is at a distance of 2 from R. But the merging of Q does not create problem because it has item A→ab• also. It is relaxation 3(C).

## 5. CONCLUSION

The merging scheme presented in highly restrictive because after merging a pair of states error in merging another pair of states may not be detected. Hence it may not be possible to reduce the number of states by more then one in a LR parser. Another disadvantage of present scheme is that merging process can be started only when complete LR parser is made. In LALR state merging starts during parser construction. In grammar S⟶ApAqg A⟶abc states with items A⟶a•bc(p) and A⟶a•bc(q) can not be merged because their merging will cause non determinism (violation of rule 1). However states with items A⟶abc•(p) and A⟶abc•(q) can be merged. After merging them states with items A⟶ab•c(p) and A⟶ab•c(q) can also be merged. After that states with items A⟶a•bc(p) and A⟶a•bc(q) are merged.

But the scheme has following advantages over LALR parser.

(1) All pair of states which are merged in LALR and conflict is not created can be merged in present scheme also. In grammar S⟶gAp S⟶Bq S⟶hgAr S⟶hBs A⟶ab B⟶gab the states with items {A⟶ab•(p), B⟶gab•(q)} and {A⟶ab•(r), B⟶gab•(s)} will be merged. After that states with items {A⟶a•b(p), B⟶ga•b(q)} and {A⟶a•b(r), B⟶ga•b(s)} are also merged. After that states with items {A⟶•ab(p), B⟶g•ab(q)} and {A⟶•ab(r), B⟶g•ab(s)} are merged.

(2) The state merging of LALR which causes conflict is prohibited in present merging scheme (using rule 1). In grammar S⟶gAp S⟶Bq S⟶hgAq S⟶hBs A⟶ab B⟶gab the states with items {A⟶ab•(p), B⟶ gab•(q)} and {A⟶ab•(q), B⟶gab•(s)} will not be merged. Hence subsequent state merging will not take place. In LALR parser firstly states with items {A⟶•ab(p), B⟶ g•ab(q)} and {A⟶•ab(q), B⟶g•ab(s)} are merged. Hence subsequent merging takes place. Finally the states with items {A⟶ab•(p), B⟶ gab•(q)} and {A⟶ab•(q), B⟶gab•(s)} are merged. It causes reduce-reduce conflict.

(3) The present scheme provides additional merging options. In grammar S⟶aAp S⟶Bq S⟶Ag A⟶rs B⟶ars states with items {A⟶rs•(p), B⟶ars•(q)} and {A⟶rs•(g)} can be merged (using rule3(relaxation C)). After that state with items {A⟶r•s(p), B⟶ar•s(q)} and {A⟶r•s(g)} can also merged (using rule 4). Finally states with items {A⟶•rs(p), B⟶a•rs(q)} and {A⟶•rs(g)} will be merged (again using rule 4). This merger is not possible in LALR parser. Following table shows comparison of LALR and present scheme.

| LALR | Present scheme |
|---|---|
| If two states have all items same but follow is different then they are merged. | They may or may not be merged. |
| As soon a state is created the decision about its merging with some existing state is taken. If a state is not merged at creation time then it will never be merged. | State merging process starts only when complete canonical LR parser is ready. Hence merging is backward process. |
| Two states which have any item different are never merged. | They may be merged. Even if two states with disjoint set of items can also be merged but this merging is permitted only once in a parser. |
| Power of LALR is less than that of Canonical LR parser. | The power does not reduce. |
| LALR parser generation takes less time than that canonical LR parser generation. It is because it has less number of states. | In present scheme parser generation takes more time. It is because merging process starts when canonical LR parser is ready. |
| Error occurs only during input scan. | Error can also be during reduction or because of failure in goto transition on nonterminal. |
| No assumption about the method of reduction of a handle into non terminal is made. | The merging scheme is applicable only when a handle is reduced into non terminal by removing few (size of handle) symbols from the stack. |

**Table 5.1**

## 6. ACKNOWLEDGMENTS
The author would like to thank the management committee of Hindustan University for their constant encouragement and support. The first author would like to thank Dr.Piriadarshani, Department of mathematics Hindustan University for the support and advice received from her.

## 7. REFERENCES
[1] Alfred V, Aho, Ravi Sethi, and Jeffery D.Ullaman, Compiler: Principles, Techniques and Tools, 1986, Addison-Wesley

[2] Dillip Kumar and Pawan Kumar, State Merging in LR parser, ACM SIGPLAN notices, Vol 41(4), pp 24-29, 2006

[3] Anderson T, Syntactic Analysis of LR (k) languages. PhD Thesis, University Newcastle-upon- Tyne, Northumberland, England, 1972

[4] Anderson T, Eve J, and Horning, J J, "Efficient LR(1) parsers." Acla Informatics 2 , pp 12-39, 1973

[5] Deremer F. L, "Simple LR(k) grammars " Comm. ACM 14,Vol 7, pp 453-460, 1971

[6] Floyd R. W, "Syntactic analysis and operator precedence " J. ACM 10,Vol 3, pp 316-333, 1963

[7] Lalonde W R, Lee E S, and Homing J. J, "An LALR (k) parser generator." Proc. IFIP Congress 71 TA-3, North-Holland Publishing Co., Amsterdam, the Netherlands, pp 153-157, 1971

[8] Pager D, "On the incremental approach to left- to-right parsing " Technical Report PE 238, Information Sciences Program, Univ. Hawaii, Honolulu, Hawan, 1972a

[9] Pager D, "A fast left-to-right parser for context-free grammars." Technical Report PE 240, Information Sciences Program, Univ. Hawaii, Honolulu, Hawaii, 1972b