SNS COLLEGE OF TECHNOLOGY

Coimbatore-36.
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 19CSB301&Automata Theory and Compiler Design

III YEAR/ V SEMESTER

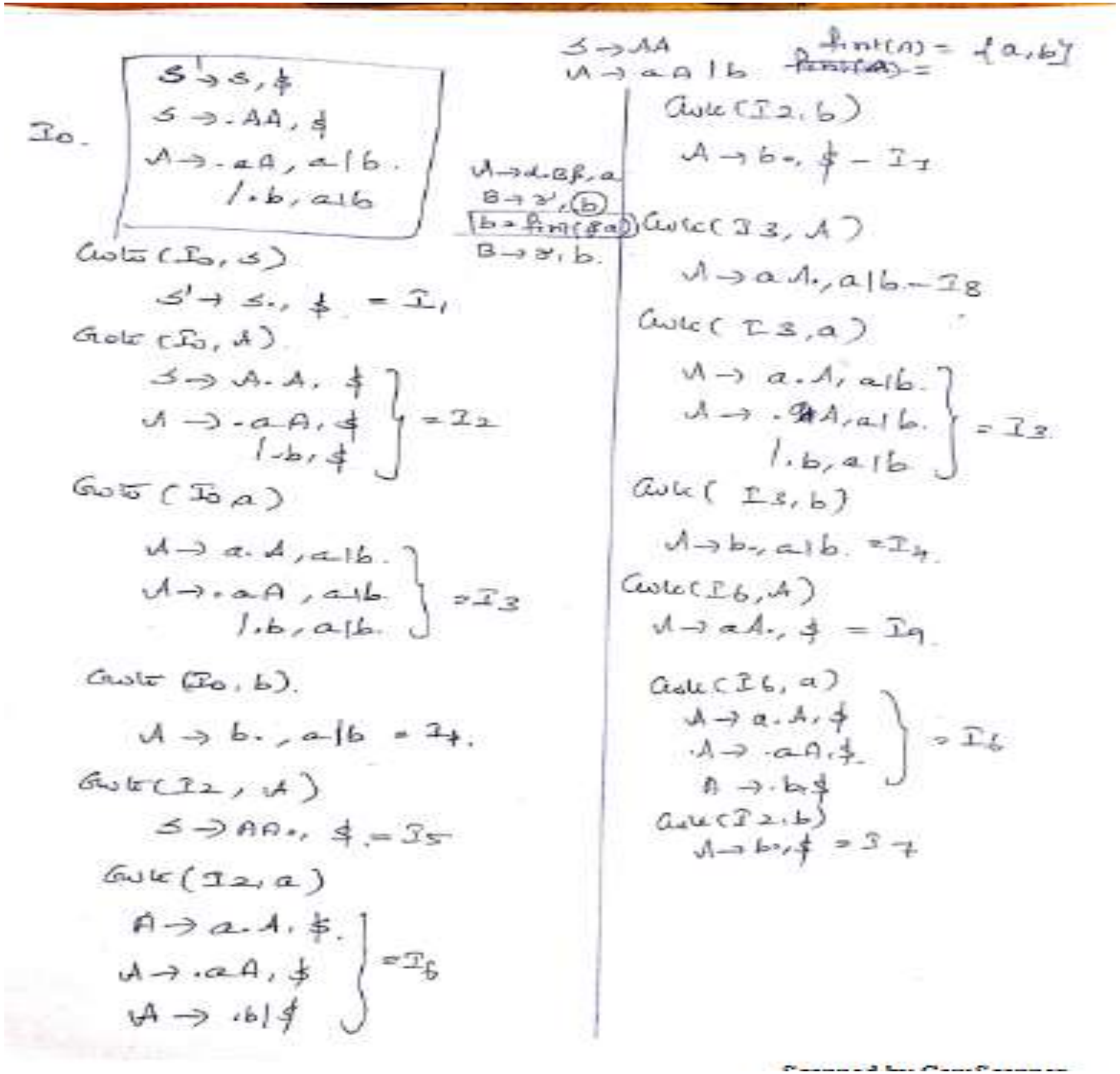
UNIT – III SYNTAX ANALYSIS AND SEMANTIC ANALYSIS

Topic: CLR &LALR

Dr.B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

CLR Parser

The CLR parser stands for canonical LR parser.It is a more powerful LR parser.It makes use of lookahead symbols. This method uses a large set of items called LR(1) items.The main difference between LR(0) and LR(1) items is that, in LR(1) items, it is possible to carry more information in a state, which will rule out useless reduction states.This extra information is incorporated into the state by the lookahead symbol. The general syntax becomes [A->∝.B, a] where A->∝.B is the production and a is a terminal or right end marker $ LR(1) items=LR(0) items + look ahead



$S \to AA$

$A \to aA \mid b$

~~$first(A) =$~~ $first(A) = \{a, b\}$

$I_0.$

$S' \to S, \$$

$S \to .AA, \$$

$A \to .aA, a|b.$

$\quad | .b, a|b$

$A \to \alpha.B\beta, a$

$B \to \gamma, \textcircled{b}$

$\boxed{b = first(\beta a)}$

$B \to \gamma, b.$

Goto (I_0, S)

$S' \to S., \$ = I_1$

Goto (I_0, A)

$$\left.\begin{array}{l} S \to A.A, \$ \\ A \to .aA, \$ \\ \quad | .b, \$ \end{array}\right\} = I_2$$

Goto $(I_0\ a)$

$$\left.\begin{array}{l} A \to a.A, a|b. \\ A \to .aA, a|b \\ \quad | .b, a|b. \end{array}\right\} = I_3$$

Goto $(I_0, b).$

$A \to b., a|b = I_4.$

Goto (I_2, A)

$S \to AA., \$ = I_5$

Goto (I_2, a)

$$\left.\begin{array}{l} A \to a.A, \$. \\ A \to .aA, \$ \\ A \to .b|\$ \end{array}\right\} = I_6$$

Goto (I_2, b)

$A \to b., \$ - I_7$

Goto (I_3, A)

$A \to aA., a|b - I_8$

Goto (I_3, a)

$$\left.\begin{array}{l} A \to a.A, a|b. \\ A \to .aA, a|b. \\ \quad | .b, a|b \end{array}\right\} = I_3$$

Goto (I_3, b)

$A \to b., a|b. = I_4.$

Goto (I_6, A)

$A \to aA., \$ = I_9$

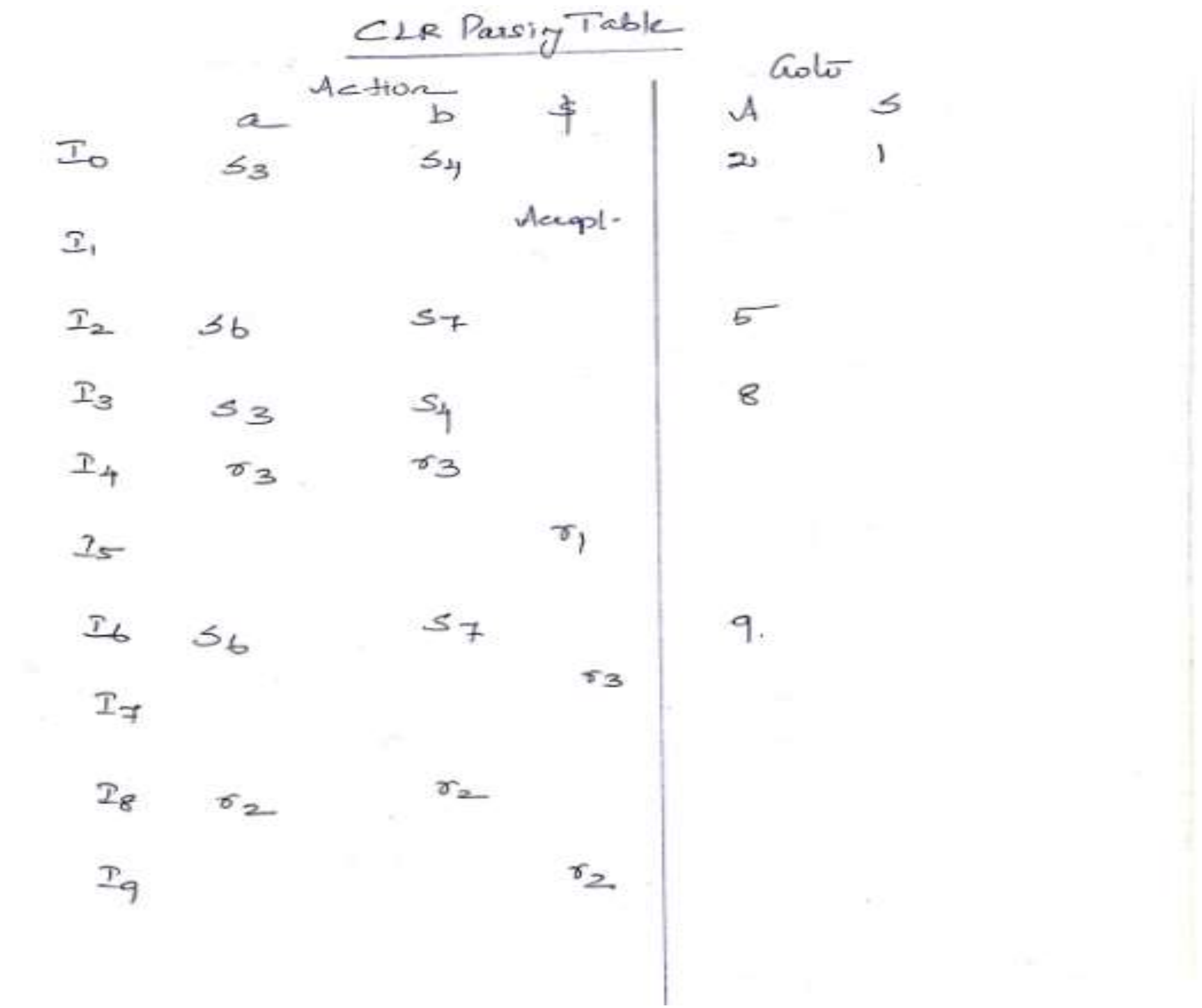
Goto (I_6, a)

$$\left.\begin{array}{l} A \to a.A, \$ \\ A \to .aA, \$ \\ A \to .b, \$ \end{array}\right\} = I_6$$

Goto (I_2, b)

$A \to b., \$ = I_7$

CLR PARSING TABLE



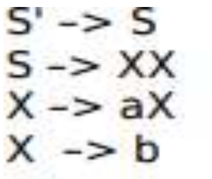
CLR Parsing Table

	Action			Goto	
	a	b	$	A	S
I0	s3	s4		2	1
I1			Accept		
I2	s6	s7		5	
I3	s3	s4		8	
I4	r3	r3			
I5			r1		
I6	s6	s7		9.	
I7			r3		
I8	r2	r2			
I9			r2		

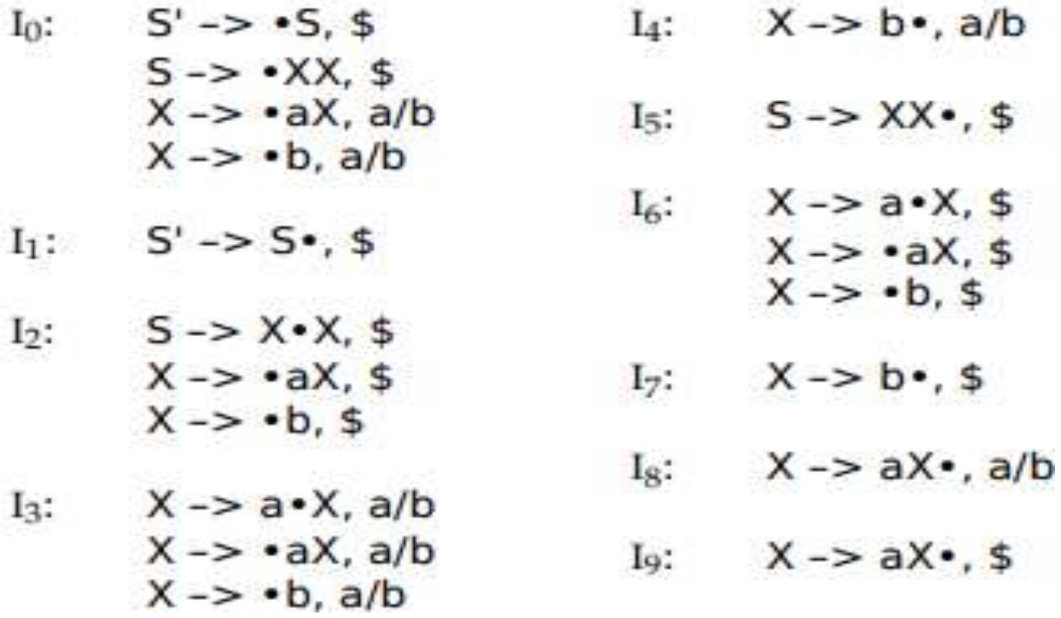
LALR PARSER

With LALR (lookahead LR) parsing, we attempt to reduce the number of states in an LR(1) parser by merging similar states. This reduces the number of states to the same a SLR(1), but still retains some of the power of the LR(1) lookaheads.

Example:



S' -> S
S -> XX
X -> aX
X -> b



I_0: S' -> •S, $
S -> •XX, $
X -> •aX, a/b
X -> •b, a/b

I_1: S' -> S•, $

I_2: S -> X•X, $
X -> •aX, $
X -> •b, $

I_3: X -> a•X, a/b
X -> •aX, a/b
X -> •b, a/b

I_4: X -> b•, a/b

I_5: S -> XX•, $

I_6: X -> a•X, $
X -> •aX, $
X -> •b, $

I_7: X -> b•, $

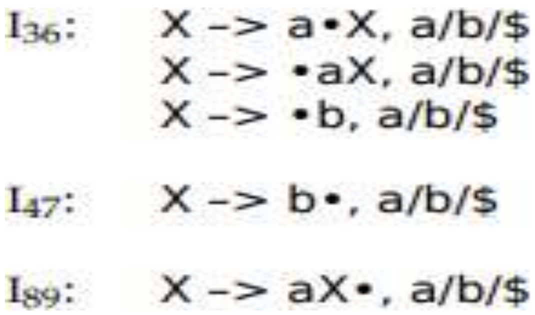
I_8: X -> aX•, a/b

I_9: X -> aX•, $

LALR PARSER

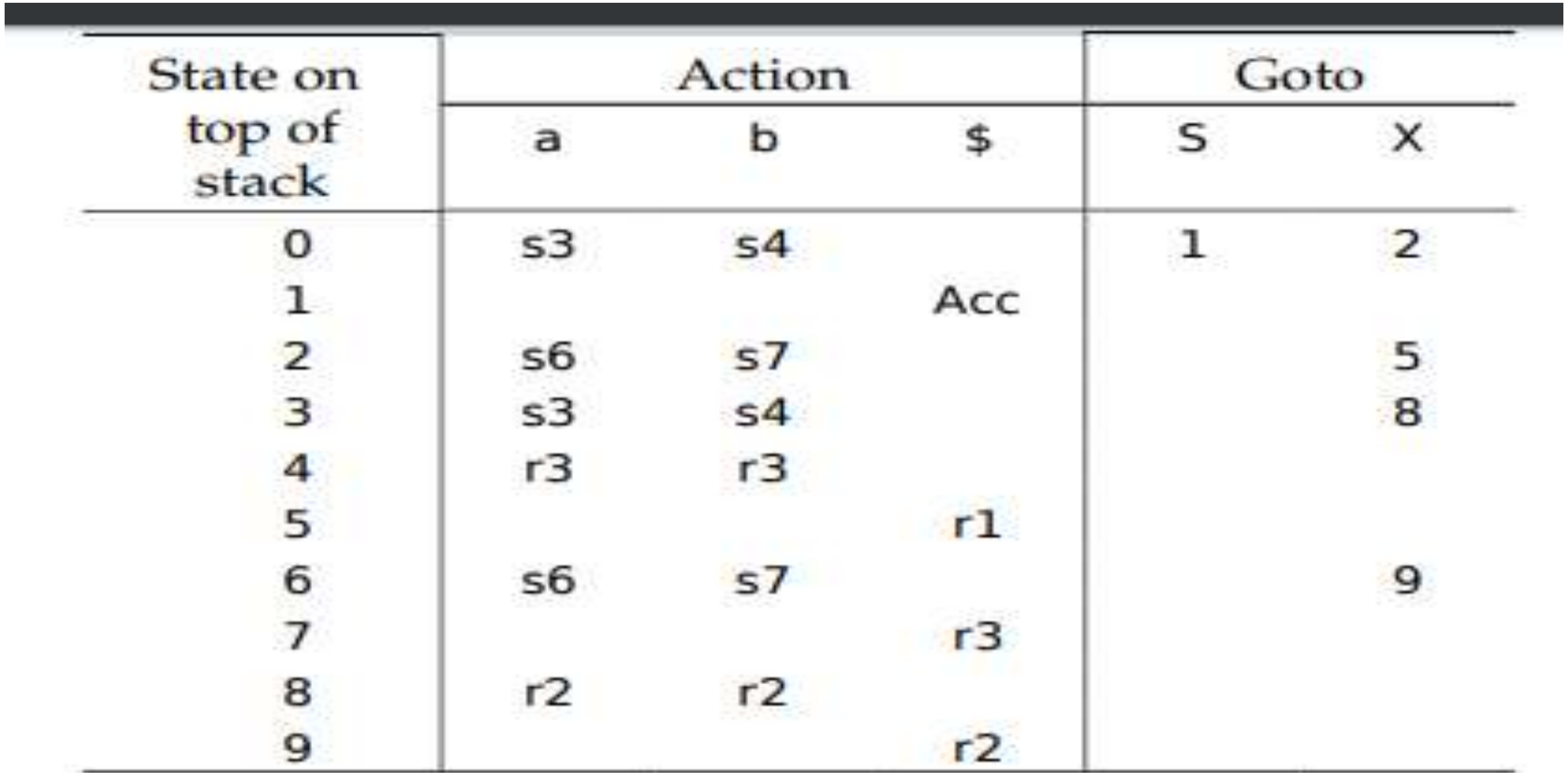
Take I3 and I6 for example. These two states are virtually identical they have the same number of items,the core of each item is identical, and they differ only in their look ahead sets

The same is true of I4 and I7, and I8 and I9. If we did merge, we would end up replacing those six states with just these three:



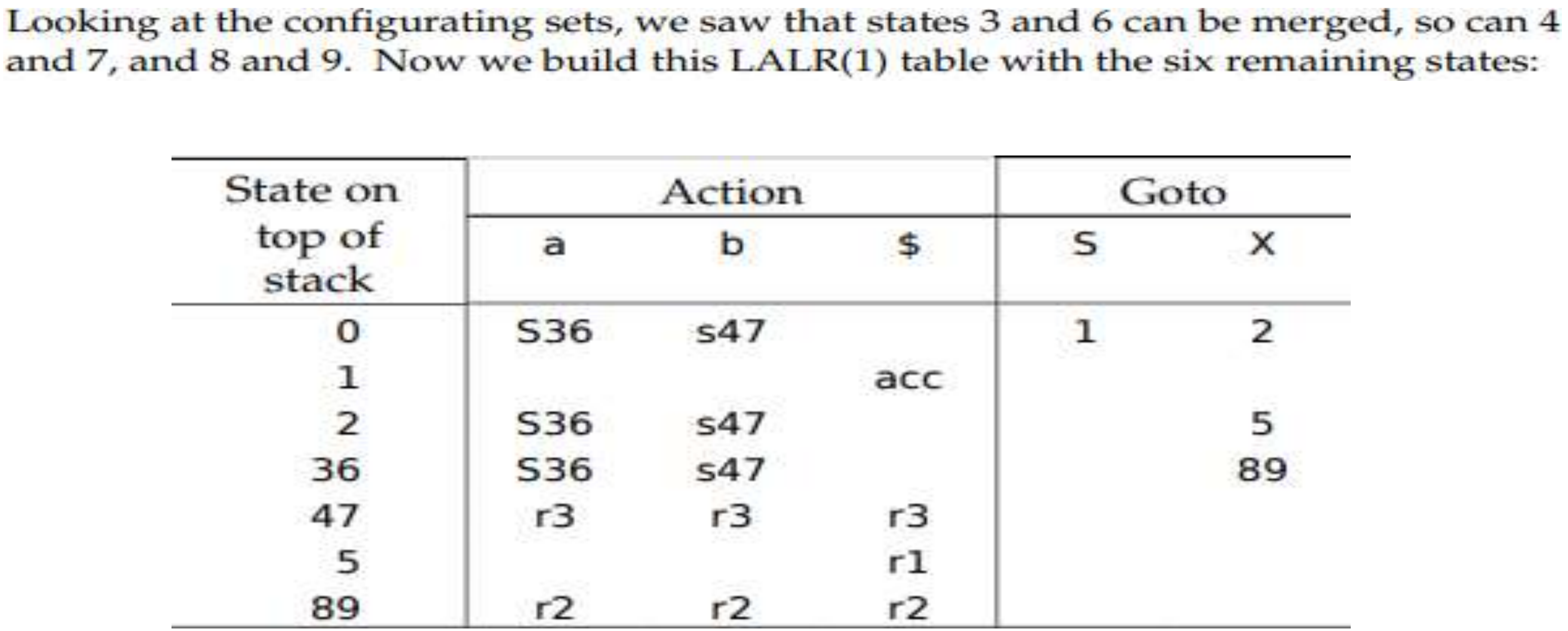
```
I36:   X -> a•X, a/b/$
       X -> •aX, a/b/$
       X -> •b, a/b/$

I47:   X -> b•, a/b/$

I89:   X -> aX•, a/b/$
```

LALR PARSER



State on top of stack	Action			Goto	
	a	b	$	S	X
0	s3	s4		1	2
1			Acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

LALR PARSER



Looking at the configurating sets, we saw that states 3 and 6 can be merged, so can 4 and 7, and 8 and 9. Now we build this LALR(1) table with the six remaining states:

State on top of stack	Action			Goto	
	a	b	$	S	X
0	S36	s47		1	2
1			acc		
2	S36	s47			5
36	S36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

Thank you