CS4031
Compiler Construction
Lecture 7

Mahzaib Younas

Lecturer, Department of Computer Science

FAST NUCES CFD

SLR Parser

- The SLR(1) parser is a simple LR parser, which is easy to construct. This is better than LR(0) as it uses a look ahead symbol.
- The “1” in SLR(1) indicates the number of lookaheads used by the parser. It uses a look ahead given by the follow set.
- The procedure for SLR(1) parsing table is the same as LR(0); the only difference is in reduce entries.

SLR Parser

- To place reduce entries once again, SLR(1) uses the DFA.
- It checks if a state has the final item.
- The state that contains a final item indicates in which row the reduce entries are to be placed. This procedure is the same as LR(0).
- For example, if state Ii has a final item, we place reduce entries in row “i.” But in row “i,” finding out columns is different for SLR(1). If it is LR(0), we place under every column, but for SLR(1) it is under the columns given by the follow set

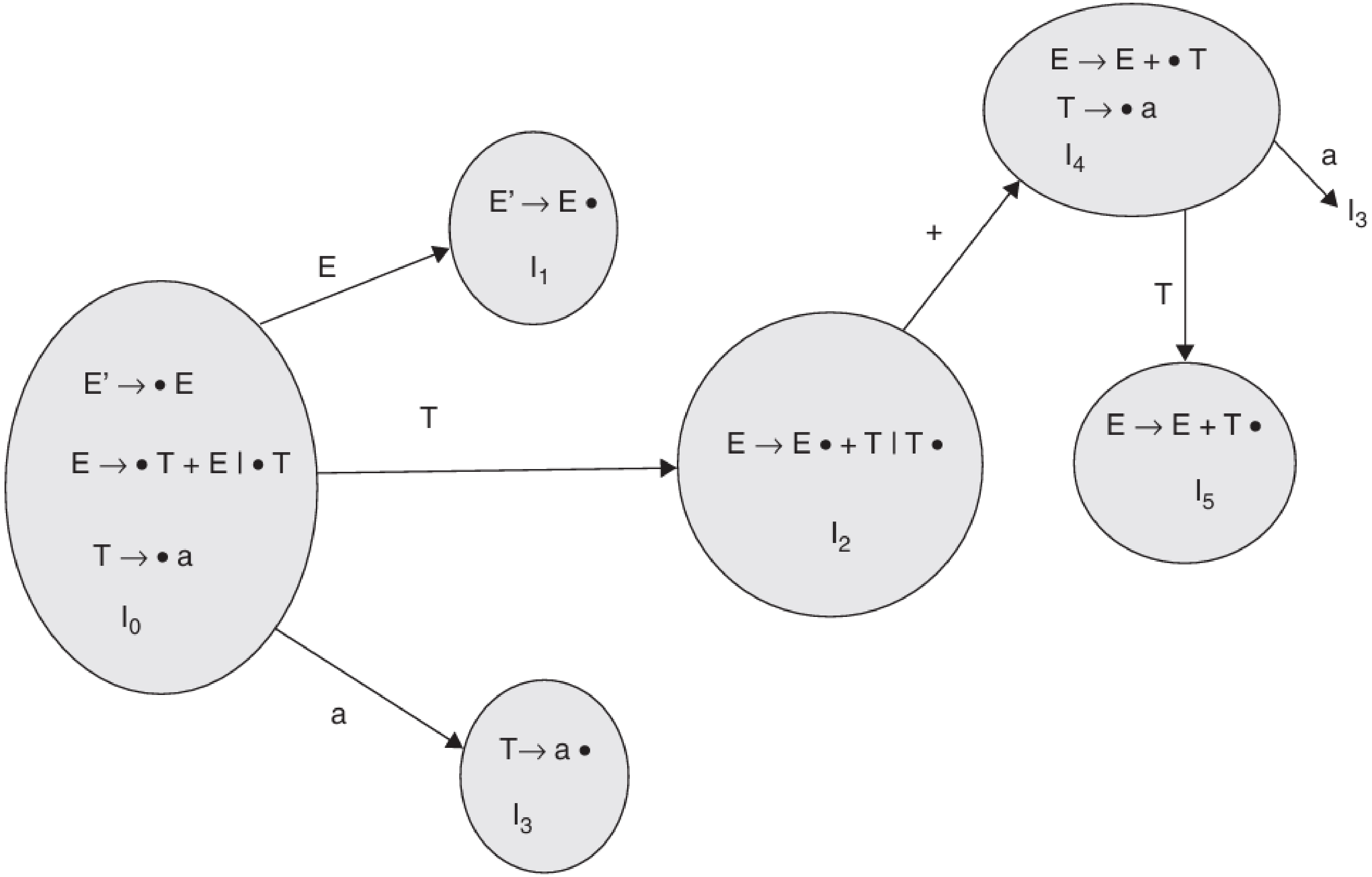
Example:

- Consider the following grammar

$E \rightarrow T + E \mid T$

$T \rightarrow a$

SLR Parser : DFA Machine



SLR Parser

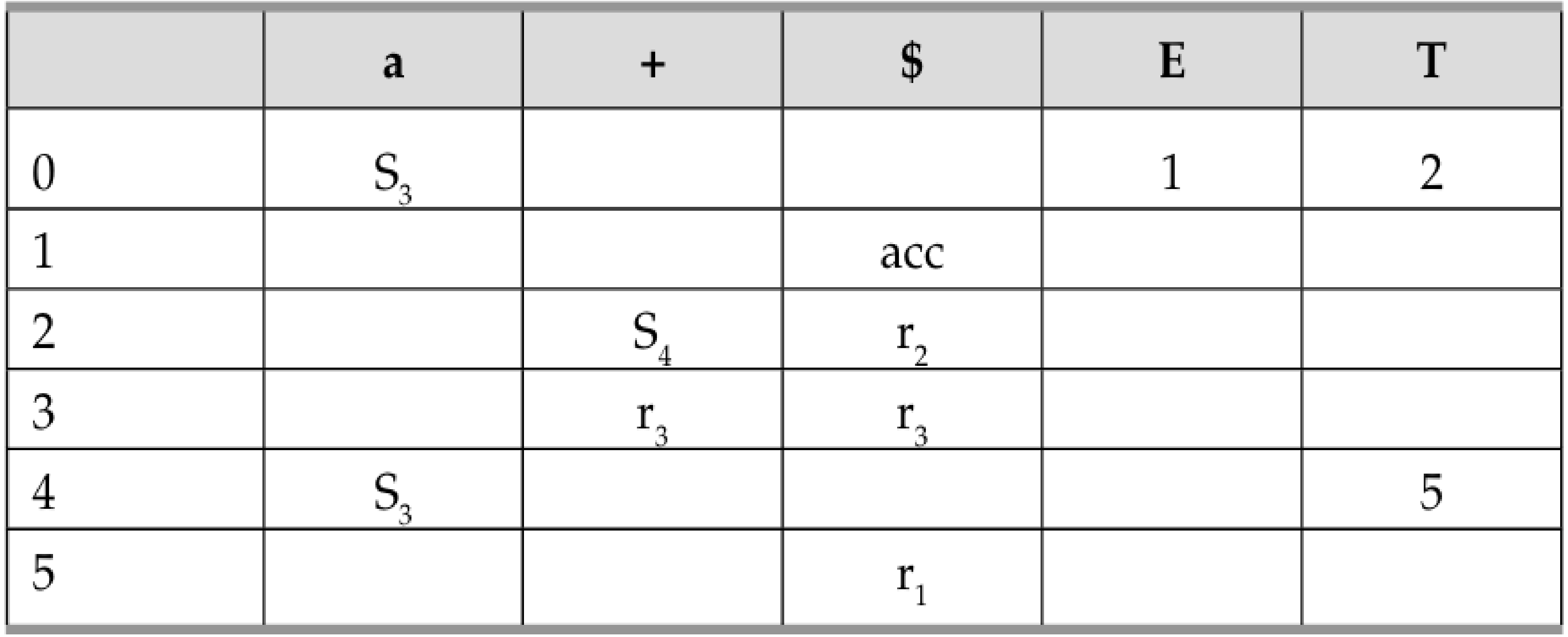
- Find the follow of the Non-Terminals

$E \rightarrow T + E \mid T$

$T \rightarrow a$

- Follow of E {$}
- Follow of T { +, $}

SLR Parser : Parsing Table

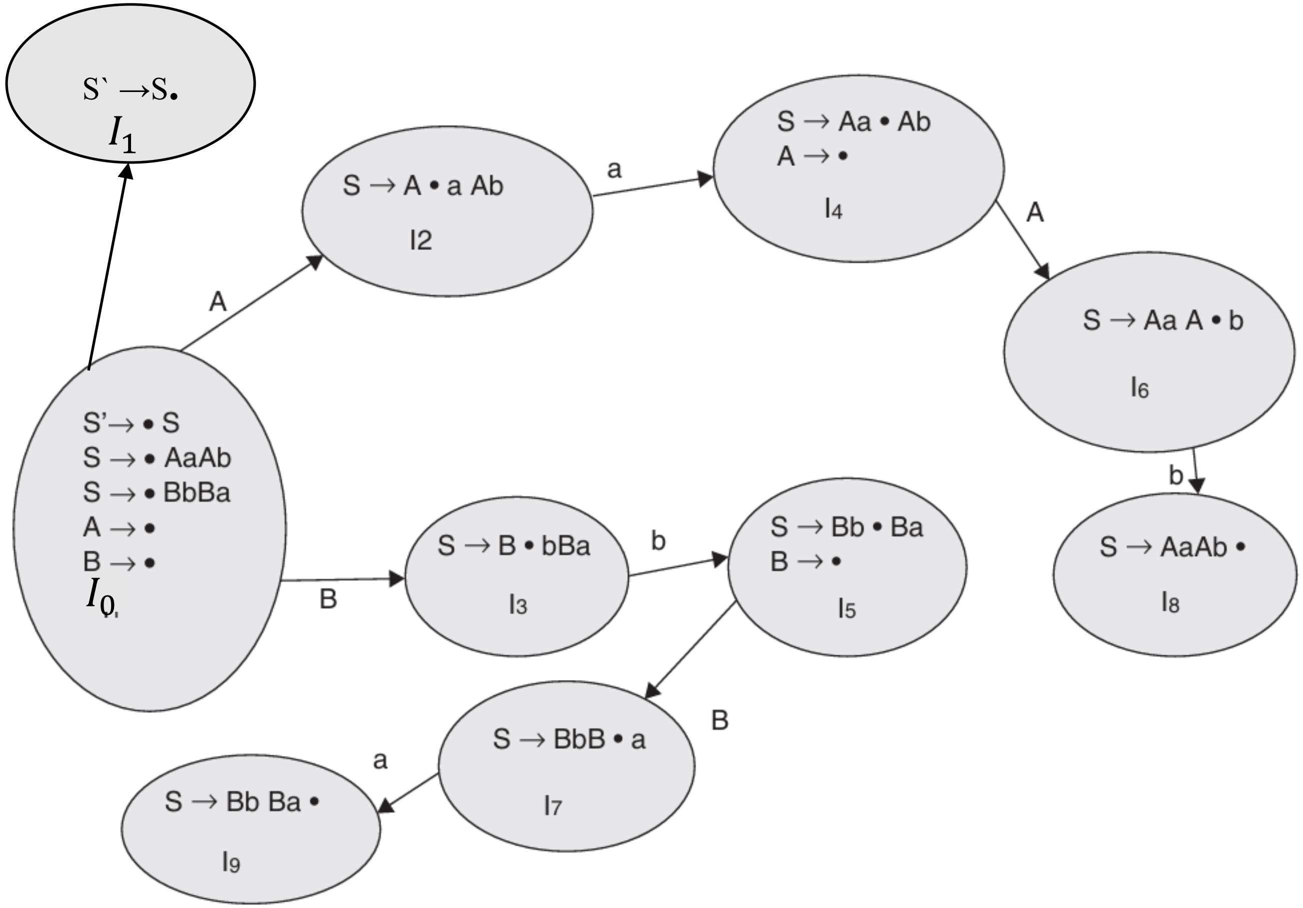


	a	+	$	E	T
0	S_3			1	2
1			acc		
2		S_4	r_2		
3		r_3	r_3		
4	S_3				5
5			r_1		

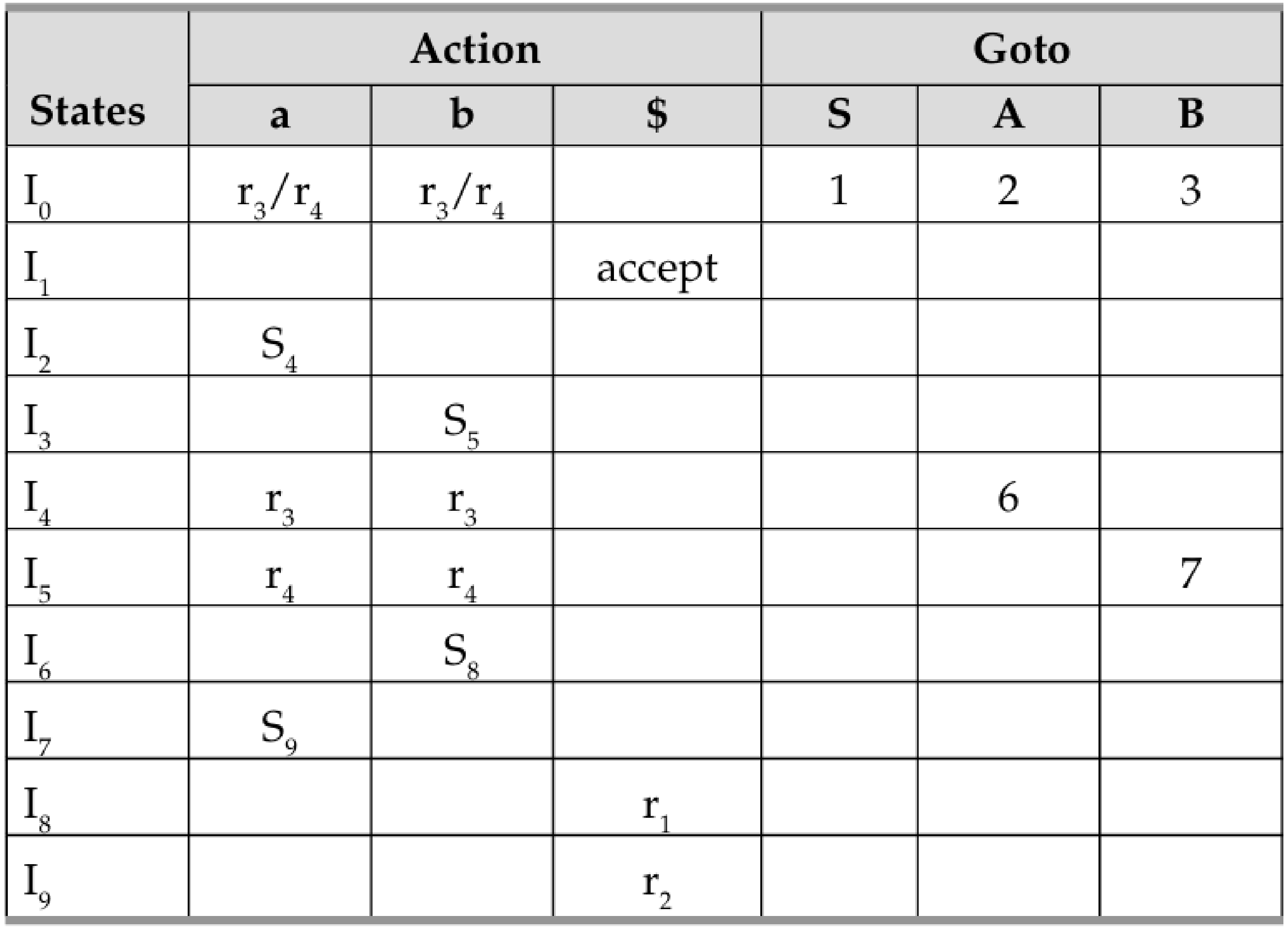
SLR Parser : Example

- Check whether the given grammar is suitable for SLR(1) parsing or not
- $S \rightarrow A\ a\ A\ b \mid B\ b\ B\ a$
- $A \rightarrow \varepsilon$
- $B \rightarrow \varepsilon$

SLR(1) Machine for the given grammar



Parsing Table



States	Action			Goto		
	a	b	$	S	A	B
I_0	r_3/r_4	r_3/r_4		1	2	3
I_1			accept			
I_2	S_4					
I_3		S_5				
I_4	r_3	r_3			6	
I_5	r_4	r_4				7
I_6		S_8				
I_7	S_9					
I_8			r_1			
I_9			r_2			

SLR (1) Example

- Consider the following grammar
- S → cAd
- A → ab | e

Augmented Grammar

- S → S’
- S → cAd
- A → ab
- A → e

SLR(1) Grammar: Parsing Table

	a	b	c	d	e	$	A	S
I_0			S_2					1
I_1								
I_2	S_4				S_5		3	
I_3				S_6				
I_4		S_7						
I_5				r_3				
I_6						r_1		
I_7				r_2				

Parsing Table : ced$

Stack	String	Action
$0	ced$	Shift s2
$0c2		

Canonical LR(1) Parsers CLR(1)/LR(1)

- 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

Need of CLR Parser

- In the SLR parser, there is a problem of shift / reduce conflict even if the grammar is unambiguous.
- This is due to the fact that the SLR parsers uses the FOLLOW() information to perform a reduce action by matching the stack information with input symbol.
- However the FOLLOW() information alone is not sufficient to decide when to reduce. Hence, powerful parser is required.

Steps of CLR Parser

The steps involved in the CALR parser are as follows:

- Construct LR(1) items – This is in contrast with the LR(0) items that is constructed for the SLR parser. This also uses Closure() and goto(), but the algorithm for these two functions are different.
- LR(1) items are used to construct the CALR parsing table involving action, goto.- The parsing table resembles SLR parsing table but has more states and there is little variation in the construction procedure.
- Use this table, along with input string and a stack is used to parse the string – The parsing action is same as the SLR parser's algorithm

CLR (1) Parser

- Consider the following grammar

S → AA

A → aA | b

CLR (1) Parser

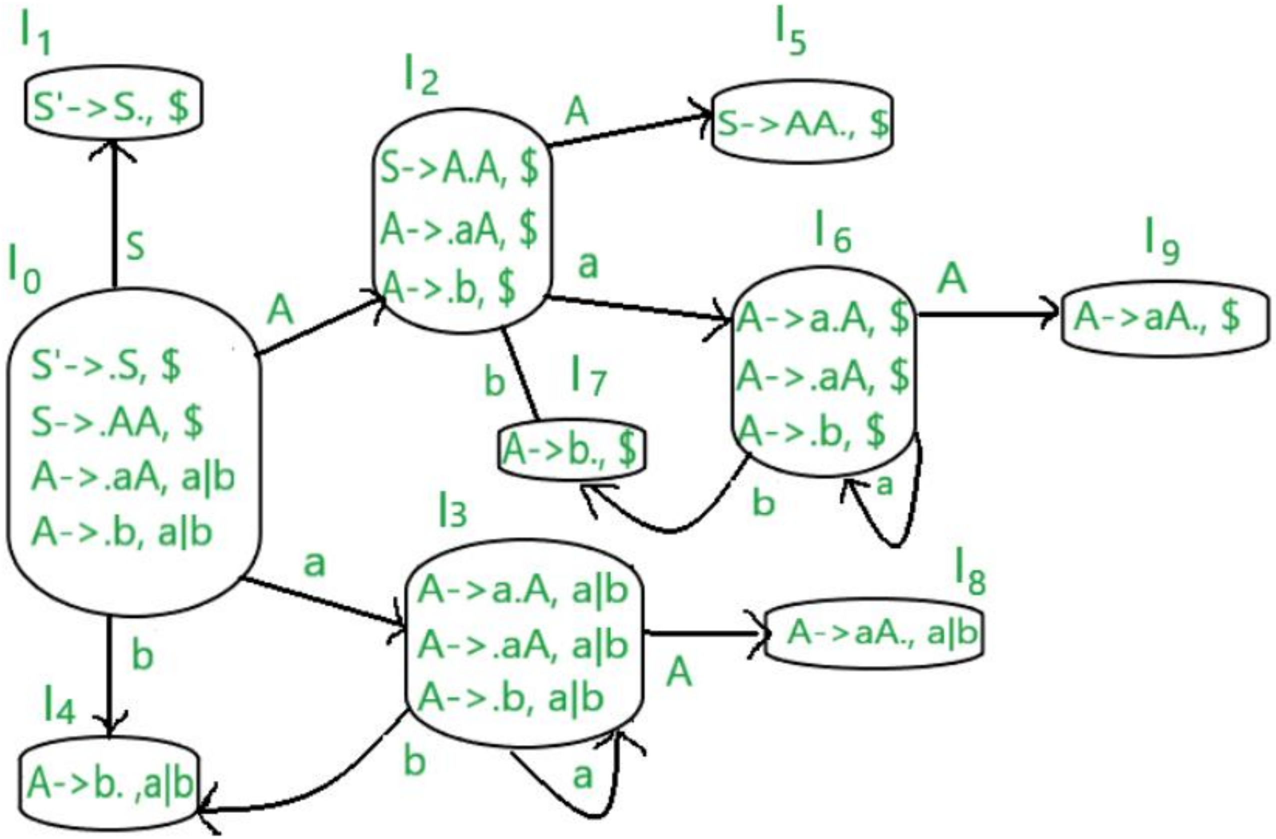
- Step 1: generate the augmented grammar

S’ → .S

S → .AA

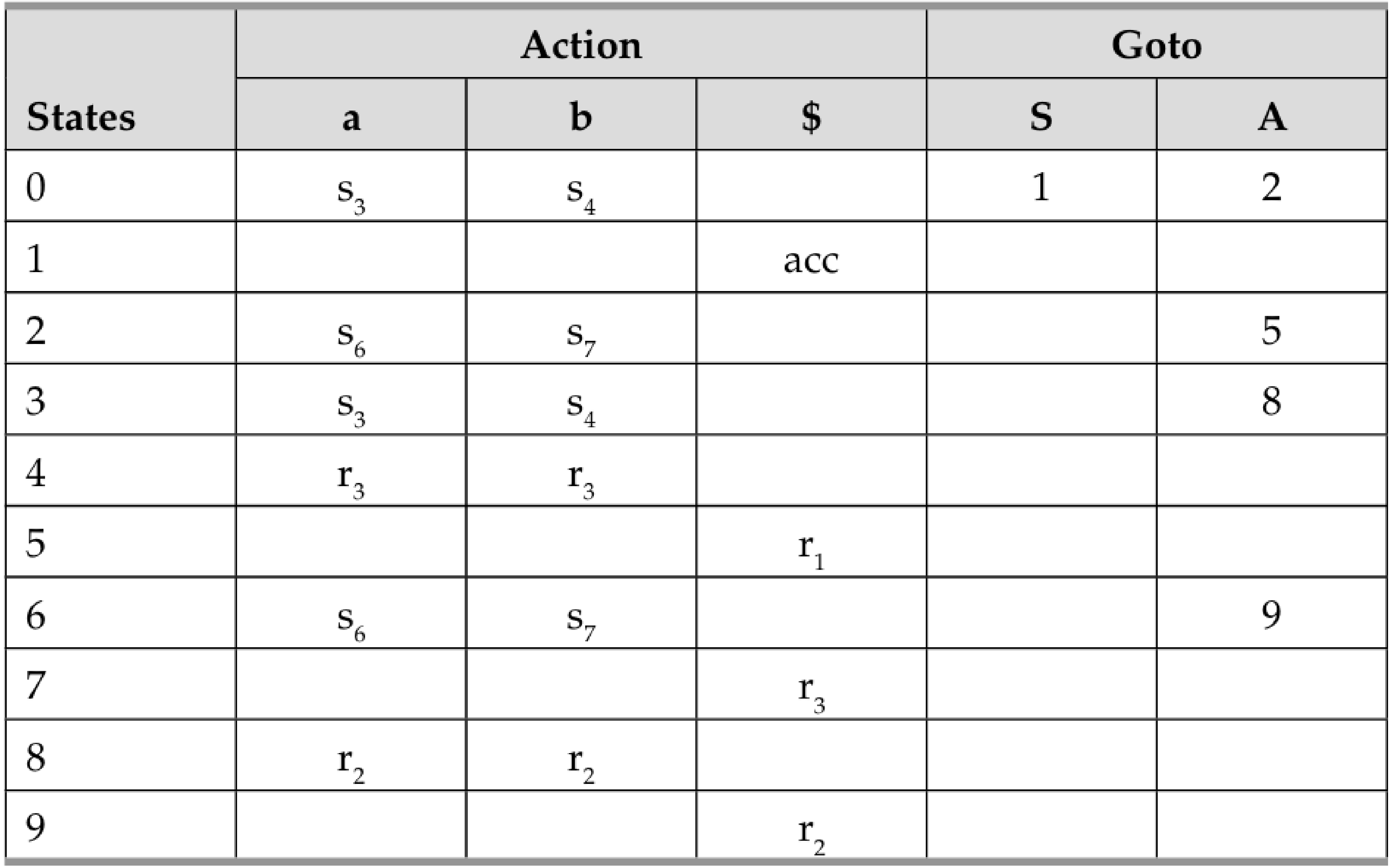
A → .aA |.b

CLR (1) Parser



I1
S'->S., $
S
I0
S'->.S, $
S->.AA, $
A->.aA, a|b
A->.b, a|b
A
I2
S->A.A, $
A->.aA, $
A->.b, $
A
I5
S->AA., $
a
I6
A->a.A, $
A->.aA, $
A->.b, $
A
I9
A->aA., $
b
I7
A->b., $
b
a
a
I3
A->a.A, a|b
A->.aA, a|b
A->.b, a|b
A
I8
A->aA., a|b
b
I4
A->b. ,a|b
b
a

CLR (1) Parsing Table



	Action			Goto	
States	a	b	$	S	A
0	s_3	s_4		1	2
1			acc		
2	s_6	s_7			5
3	s_3	s_4			8
4	r_3	r_3			
5			r_1		
6	s_6	s_7			9
7			r_3		
8	r_2	r_2			
9			r_2		

Example:

- Check whether the given grammar is suitable for CLR(1) parsing or not
- S → A a A b | B b B a
- A → ε
- B → ε