LALR(1) & CLR(1) Parsing

Baivab Das
Lecturer
Department of CSE
University of Asia Pacific

Introduction

- We have already seen LR(0) and SLR(1)
- We know how to generate the LR(0) items of a grammar
- In LR(0) we place the reduce move on the entire row
- In SLR(1) we place the reduce move only in the follow of left-hand side

LR(0) Parsing Table

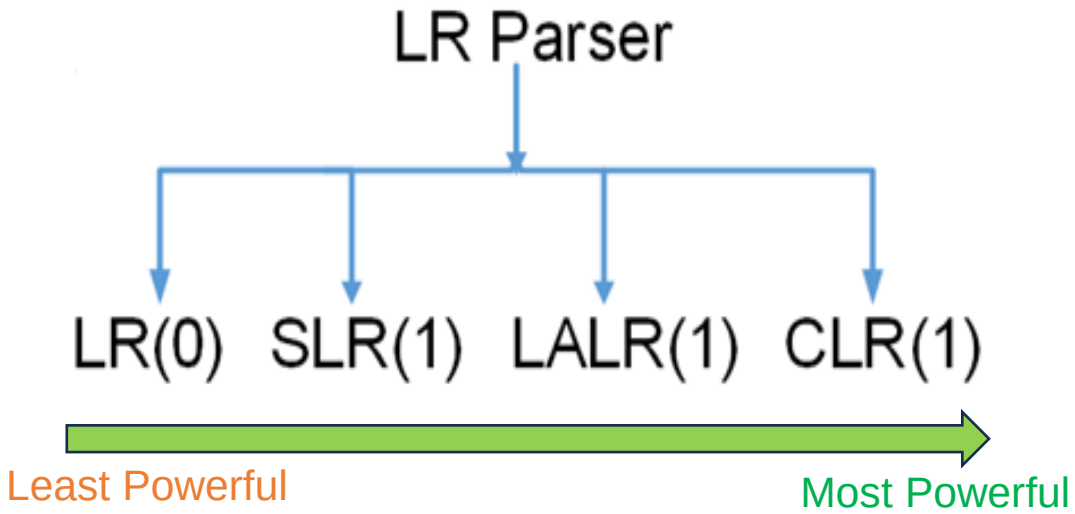
	Action			Goto	
	a	b	$	A	S
0	s_3	s_4		2	1
1			Accept		
2	s_3	s_4		5	
3	s_3	s_4		6	
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

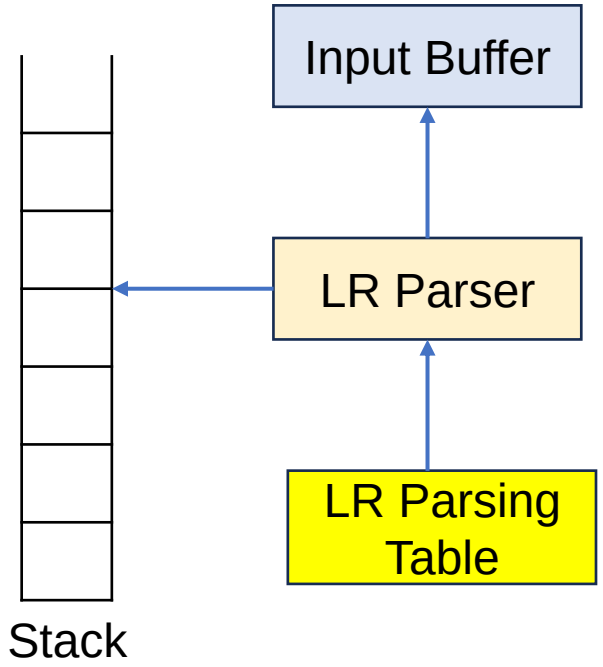
SLR(1) Parsing Table

	Action			Goto	
	a	b	$	A	S
0	s_3	s_4		2	1
1			Accept		
2	s_3	s_4		5	
3	s_3	s_4		6	
4	r_3	r_3	r_3		
5			r_1		
6	r_2	r_2	r_2		

Title Image Generated by stable diffusion realistic-vision-v51

LR Parsers

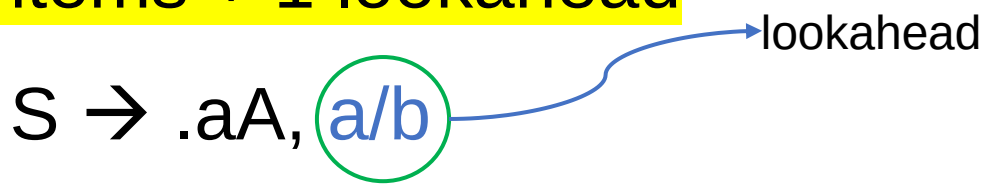




- For all the 4 parsers, the **parsing algorithm is same**
- Only change is the construction of **parsing table**
- **Canonical collection of LR(0) items** are used to construct the parsing table of LR(0) and SLR(1) parser
- **Canonical collection of LR(1) items** are used to construct the parsing table of LALR(1) and CLR(1) parser

LALR(1) & CLR(1)

- LALR(1) means Look Ahead LR and
- CLR(1) means Canonical LR
- Both of LALR(1) and CLR(1) has 1 lookahead
- So, instead of using canonical collection of LR(0) items, we use canonical collection of **LR(1) items**
- LR(1) items = LR(0) items + 1 lookahead



- Whenever we reach the final item i.e; S → aA. , a/b, **we place the reduce move in the lookahead**
- Lookaheads are useful to determine the final item

S →AA
A→aA | b

A → α.Bβ,a | b
Closure on .B
B →.γ,(lookahead = FIRST(β,a/b)

Similarly,
A → α.B, a | b
Closure on .B
B →.γ, lookahead = FIRST(a | b)

1. Start with the augmented production

S' →.S,$

($ is the lookahead for the augmented production)

2. Since there is a LR(0) item in the beginning of S, we use the CLOSURE property

S →.AA

3. As we are generating the LR(0) items, what will be the lookahead for S →.AA? .

Whatever is remaining after S in this, S' →.S,$ we calculate the first of that

We can observe that only $ is remaining after S. So, FIRST($) = {$}

S →.AA, $

Now, closure on A and remaining production is A$. So, FIRST(A$)= a/b

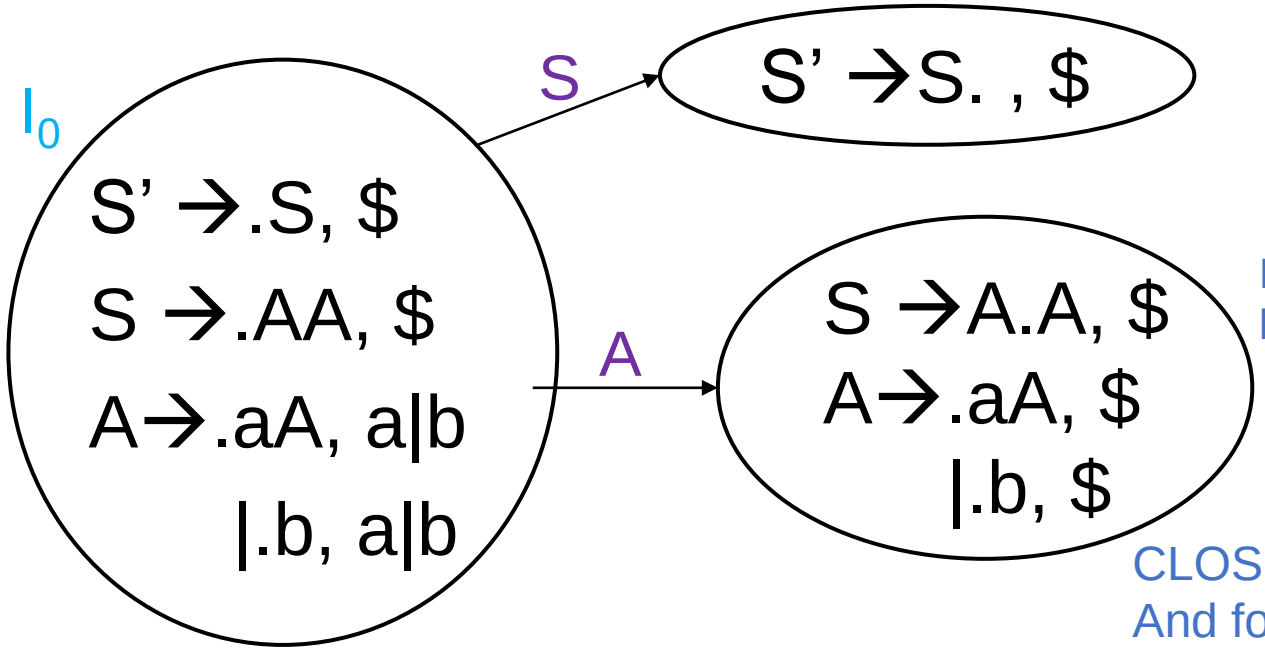
A→.aA, a/b

|.b, a/b

This is I_0

S →AA
A→aA | b

Here lookahead remains same

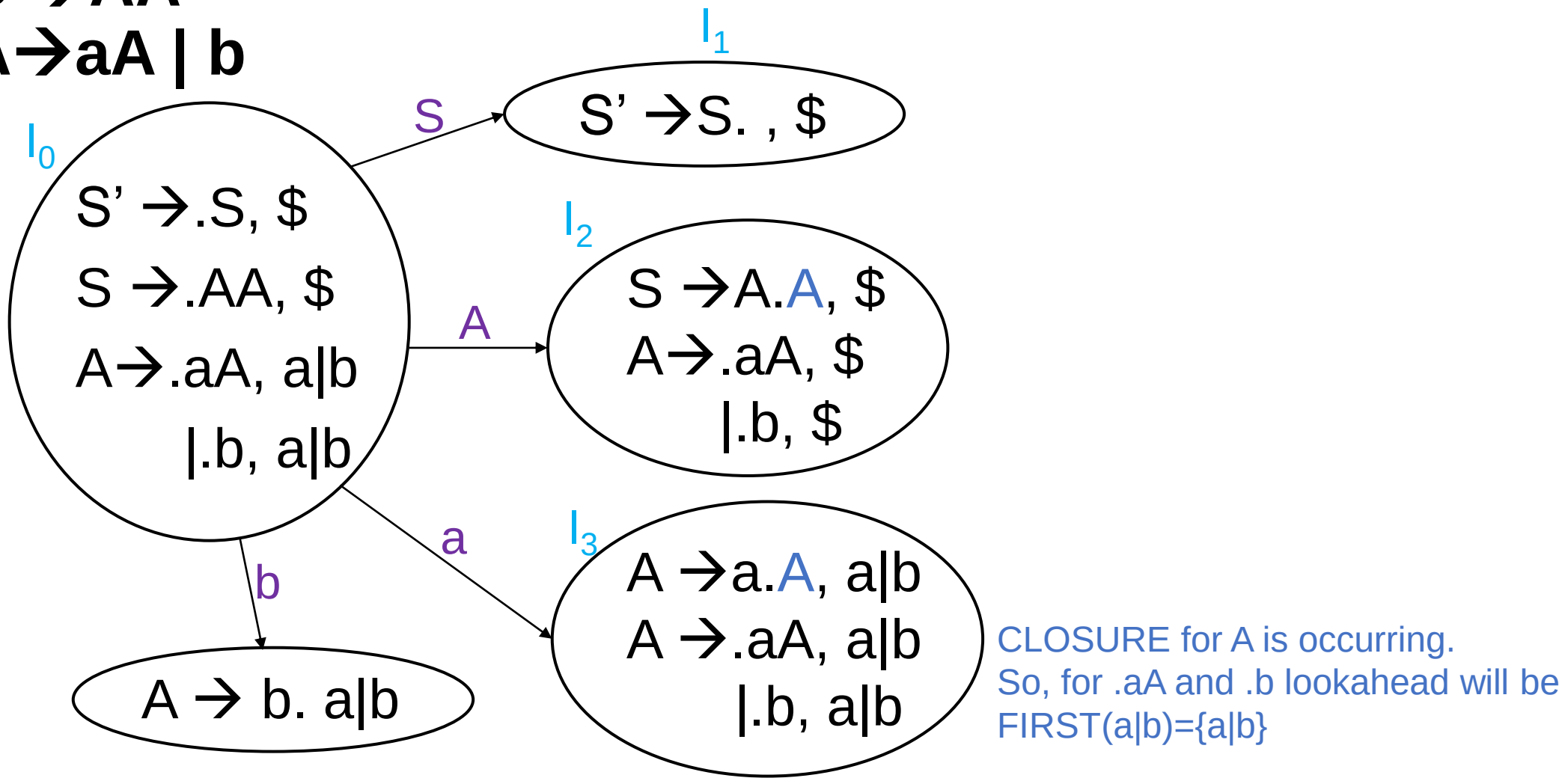


For S→A.A lookahead remains $ due but CLOSURE occurs!

CLOSURE: We write the LR(1) items for A.
And for lookahead, S→A.A,$ remaining is $.
FIRST($)={$}
So, for .aA and .b lookahead will be $

***While applying transition, the lookahead doesn’t change

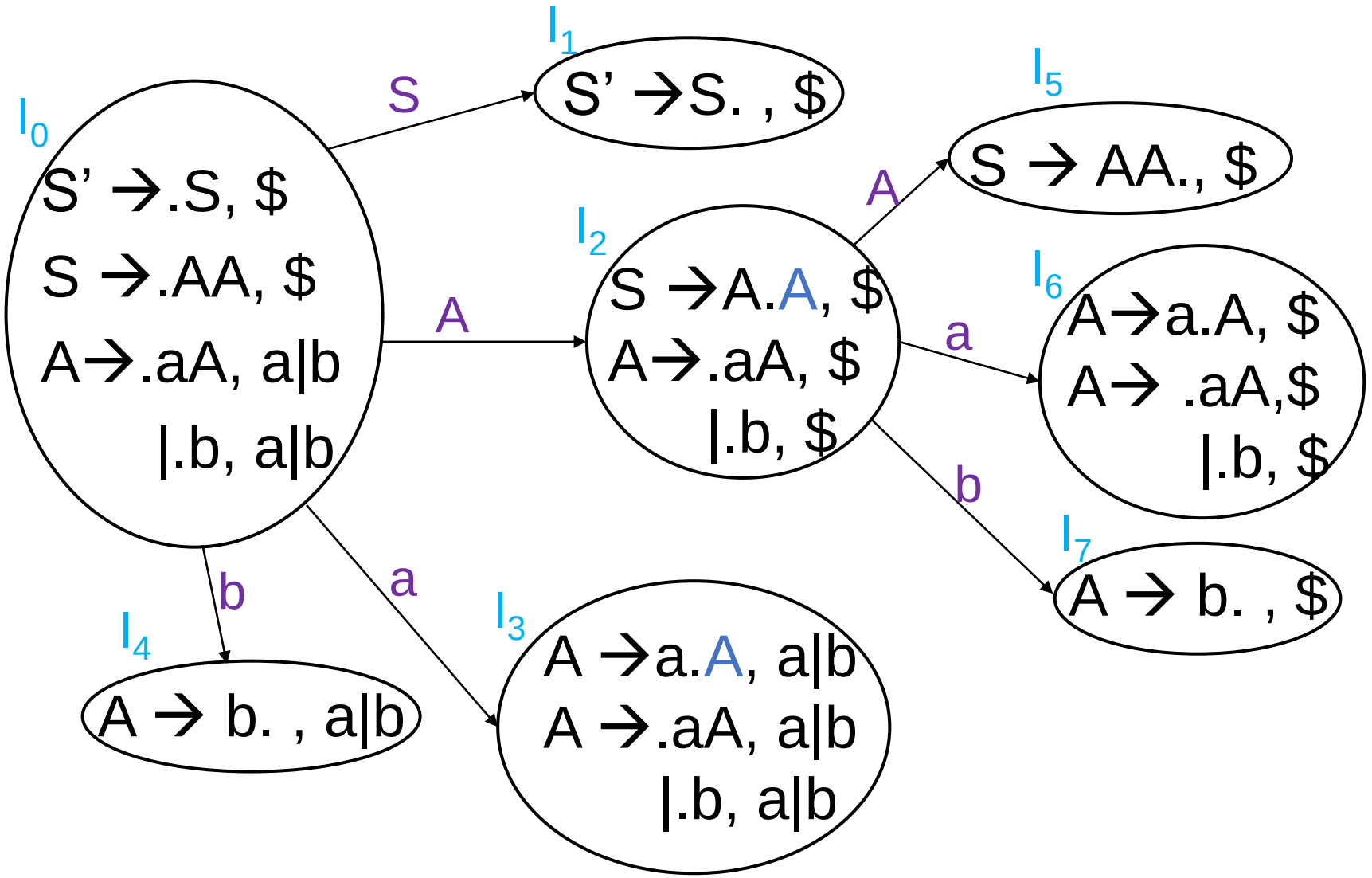
S →AA
A→aA | b



CLOSURE for A is occurring.
So, for .aA and .b lookahead will be FIRST(a|b)={a|b}

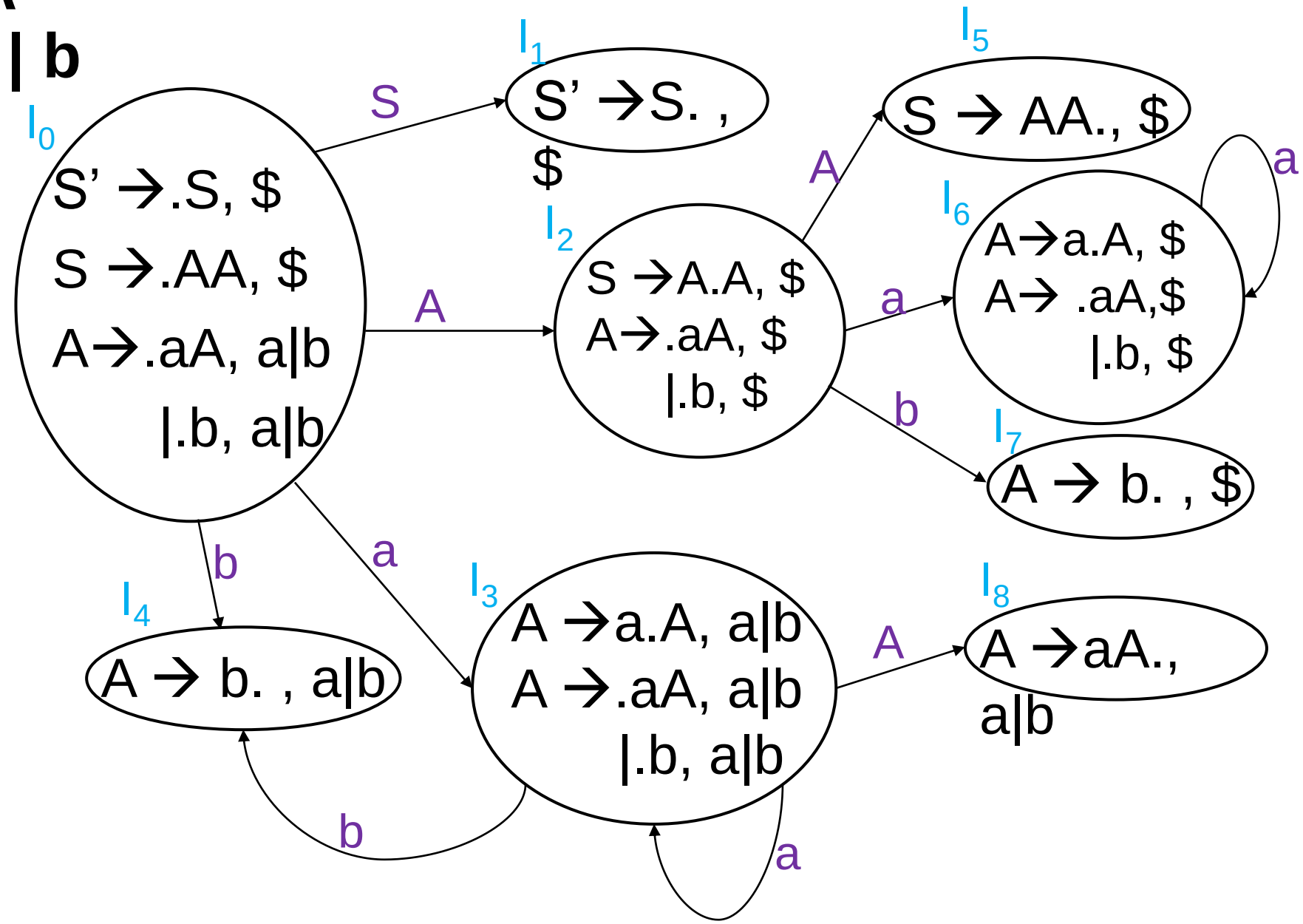
***While applying transition, the lookahead doesn't change
But, while applying closure, lookahead might change

S →AA
A→aA | b

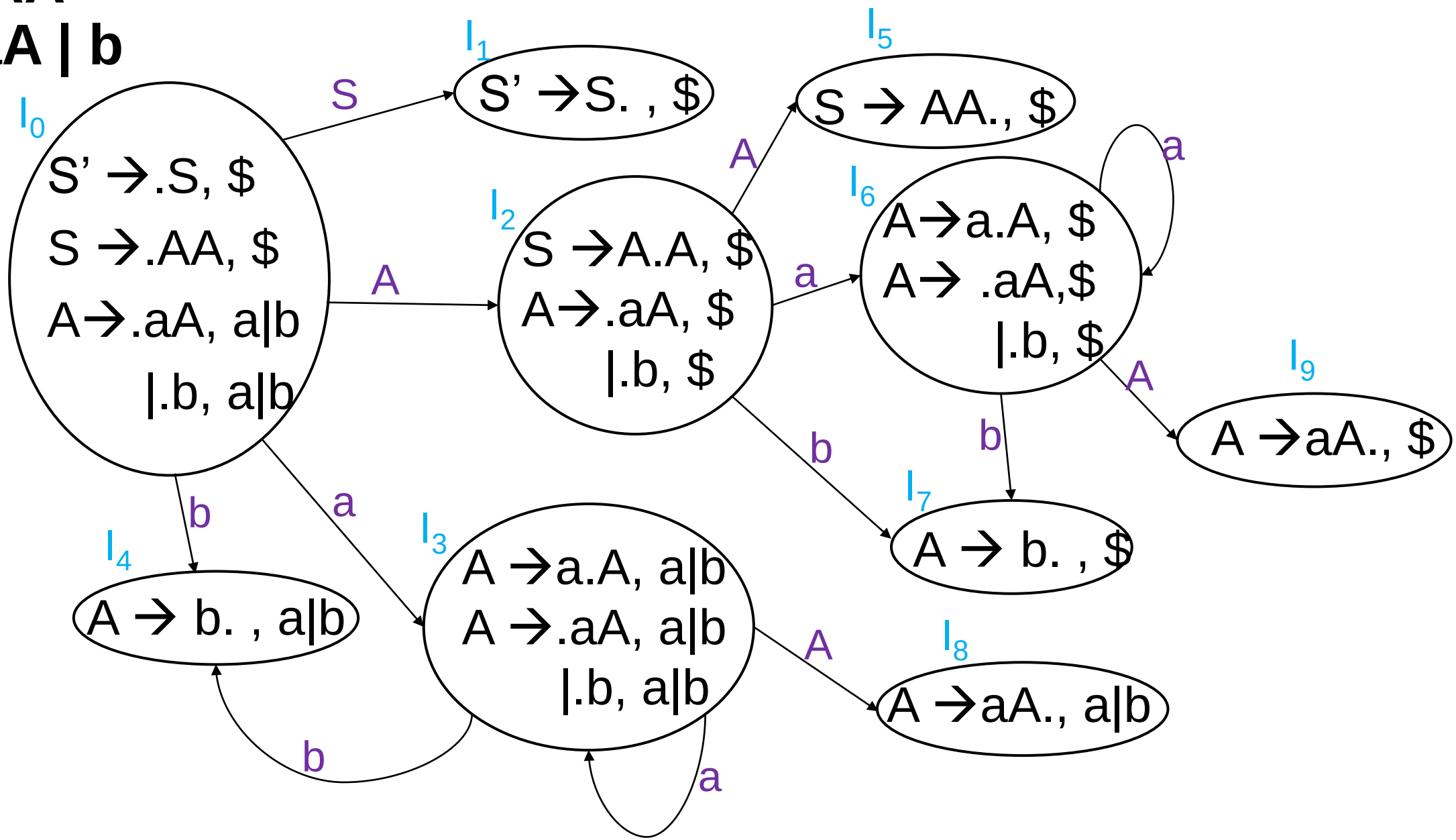


CLOSURE for A is occurring.
In terms of LR(0), I_6 andI_3 should be same.
But, in terms of lookahead there is difference.
So, for .aA and .b lookahead will be $

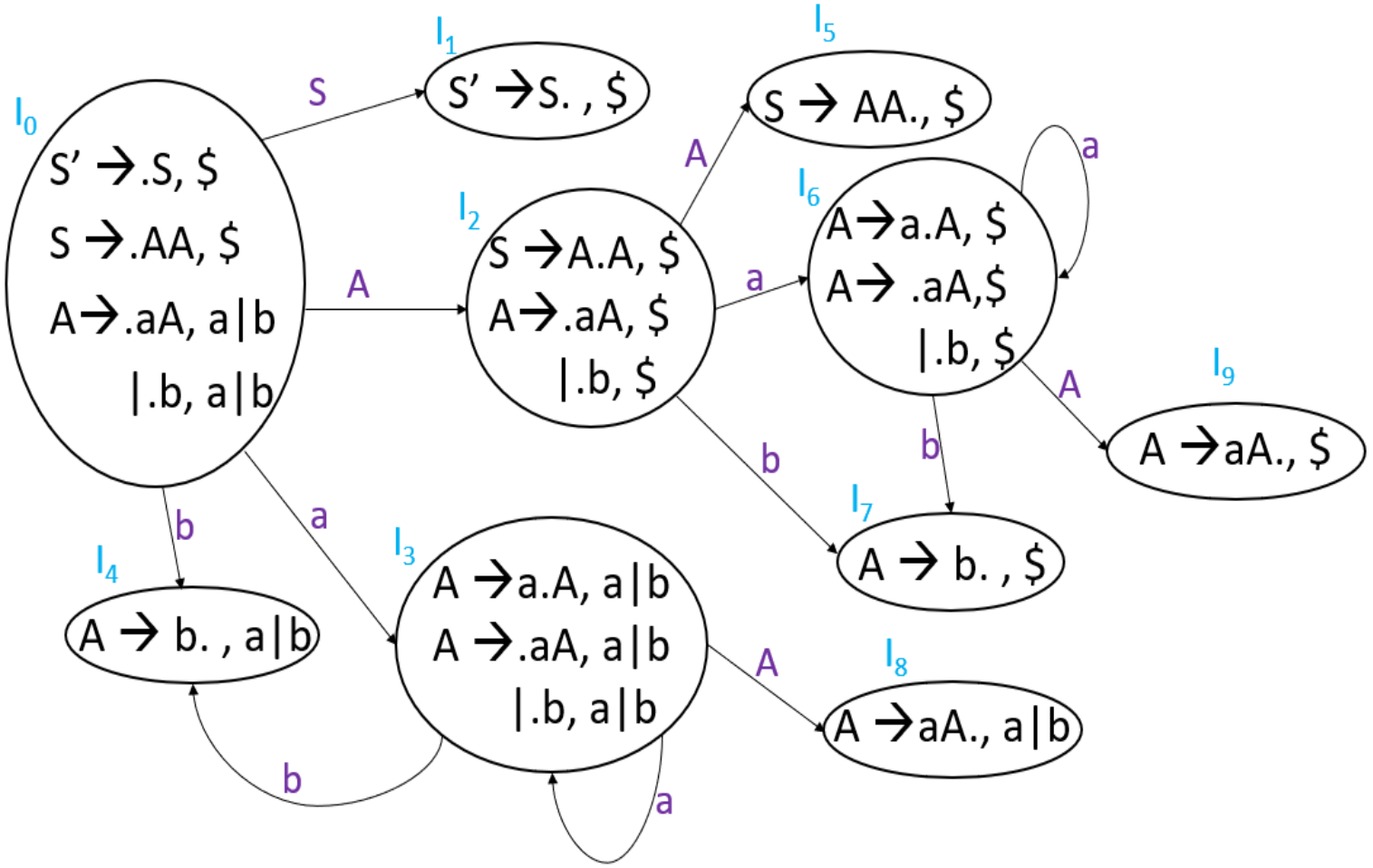
S →AA
A→aA | b


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

S →AA
A→aA | b

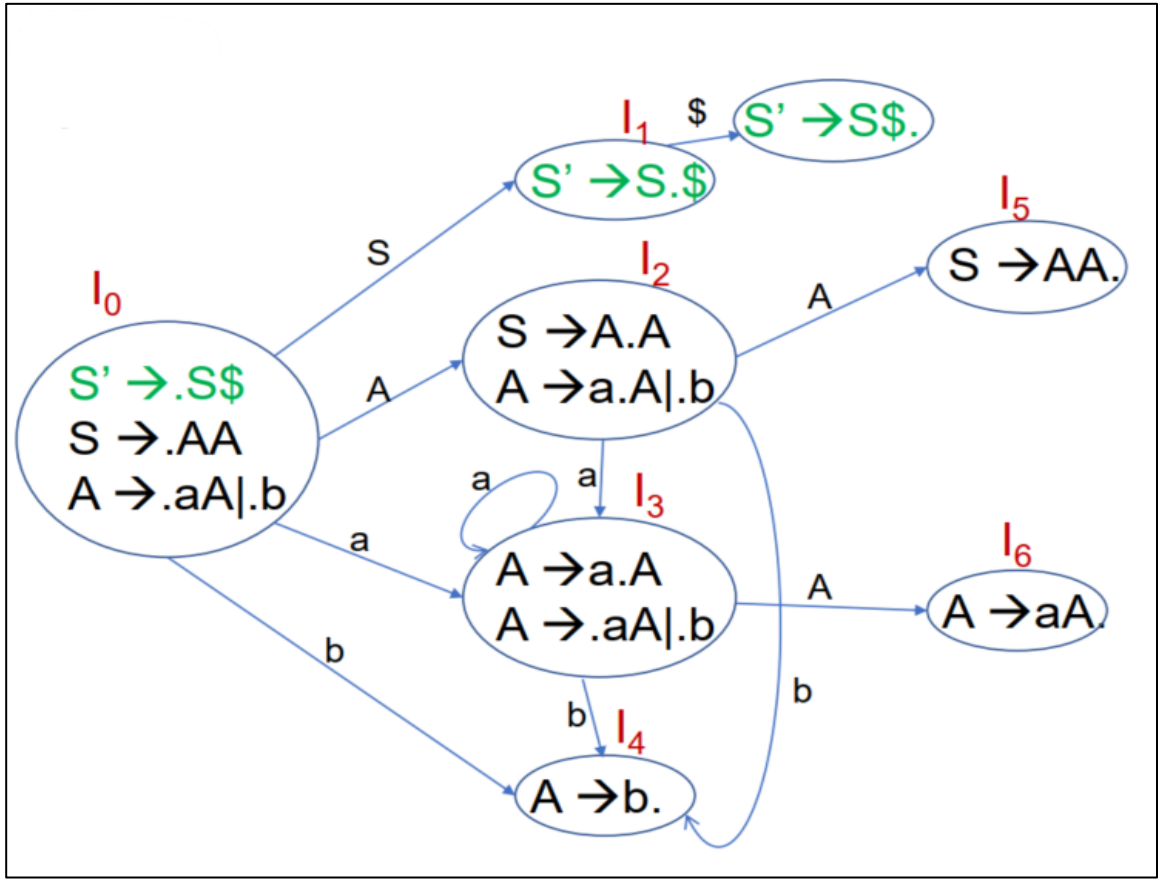

I0
S' →.S, $
S →.AA, $
A→.aA, a|b
|.b, a|b
S
I1
S' →S. , $
A
I2
S →A.A, $
A→.aA, $
|.b, $
I5
S → AA., $
a
I6
A→a.A, $
A→ .aA,$
|.b, $
A
I9
A →aA., $
b
I7
A → b. , $
a
I3
A →a.A, a|b
A →.aA, a|b
|.b, a|b
b
I4
A → b. , a|b
A
I8
A →aA., a|b
b
a

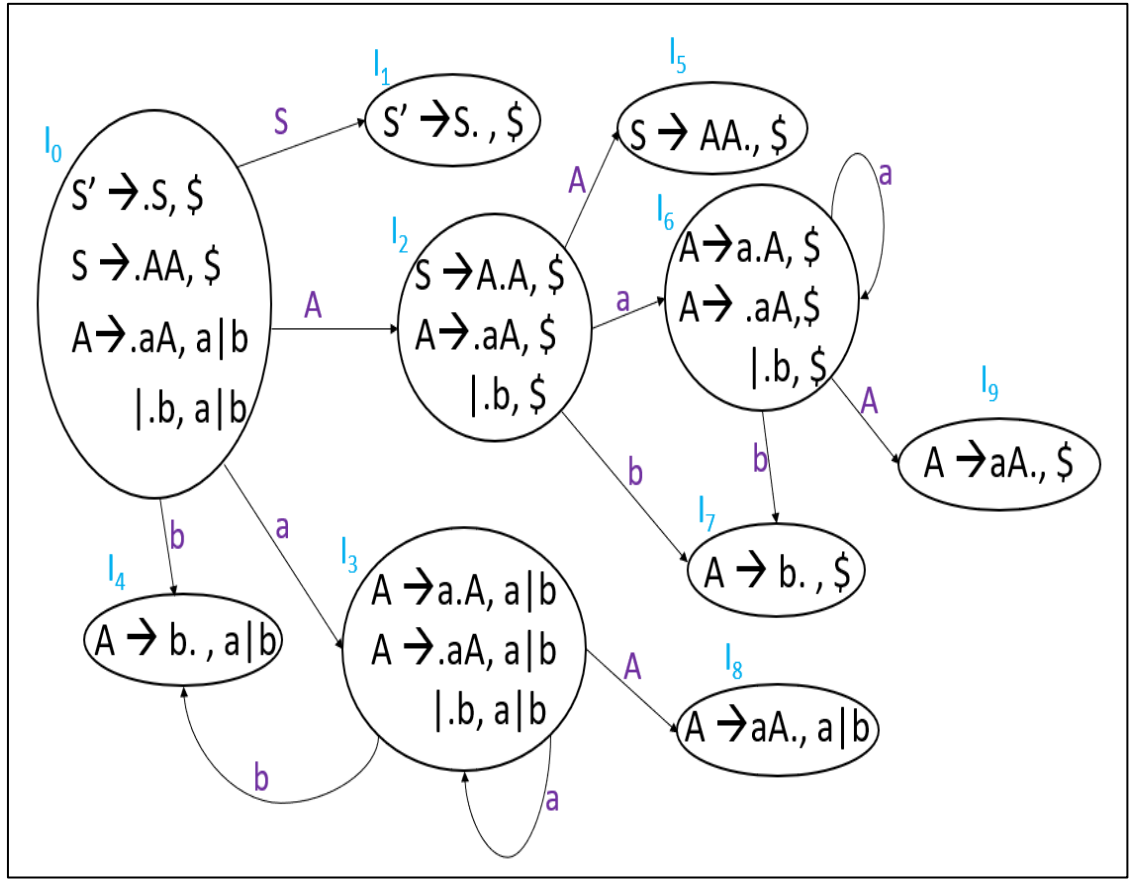
S →AA
A→aA l b



- In LR(1) items, the numbers of states has **increased** compared with LR(0) items
- In terms of LR(0) items, both I_3 and I_6 are **same**
- Also, both I_4 and I_7 are **same** in LR(0)
- Similarly, both I_8 and I_9 are **same** in LR(0)
- But, in terms of **lookahead**, **they are different**

LR(0) vs LR(1) Items



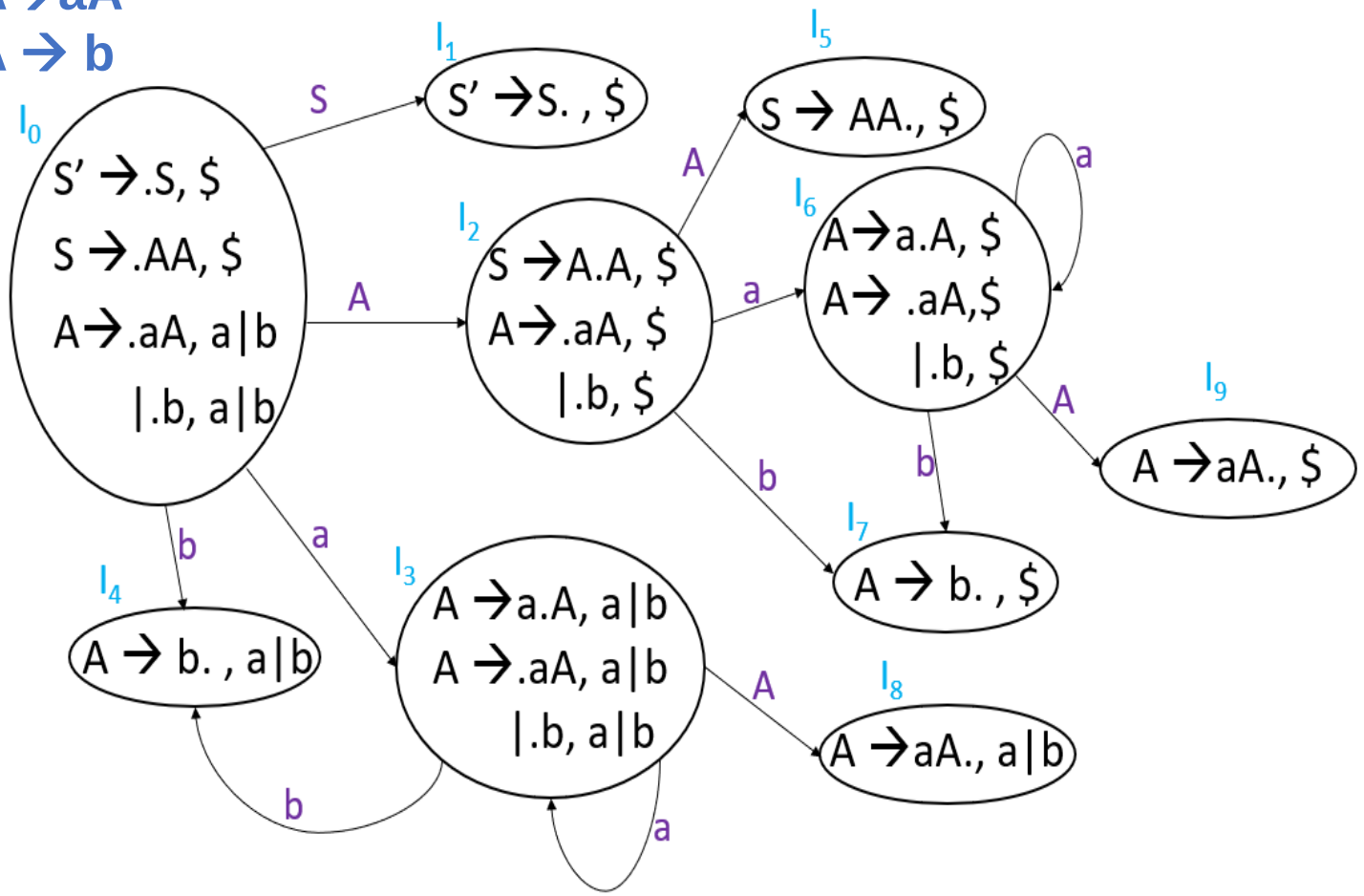


- The number of states increases in LR(1) due to different lookaheads

CLR(1) Parsing Table

- In case of CLR(1) parsing table, GOTO and SHIFT will be same as before
- **Only difference will be in placing the final item**
- In CLR(1) parsing table, we will place the **reduce move only in the lookahead symbol**
- So, the number of reduce moves are **less than SLR(1)**
- So, the possibility of **conflict** also **decreases**
- As the reduce move is decreasing, **blank spaces are also increasing**. Which means, error detecting capabilities also increases

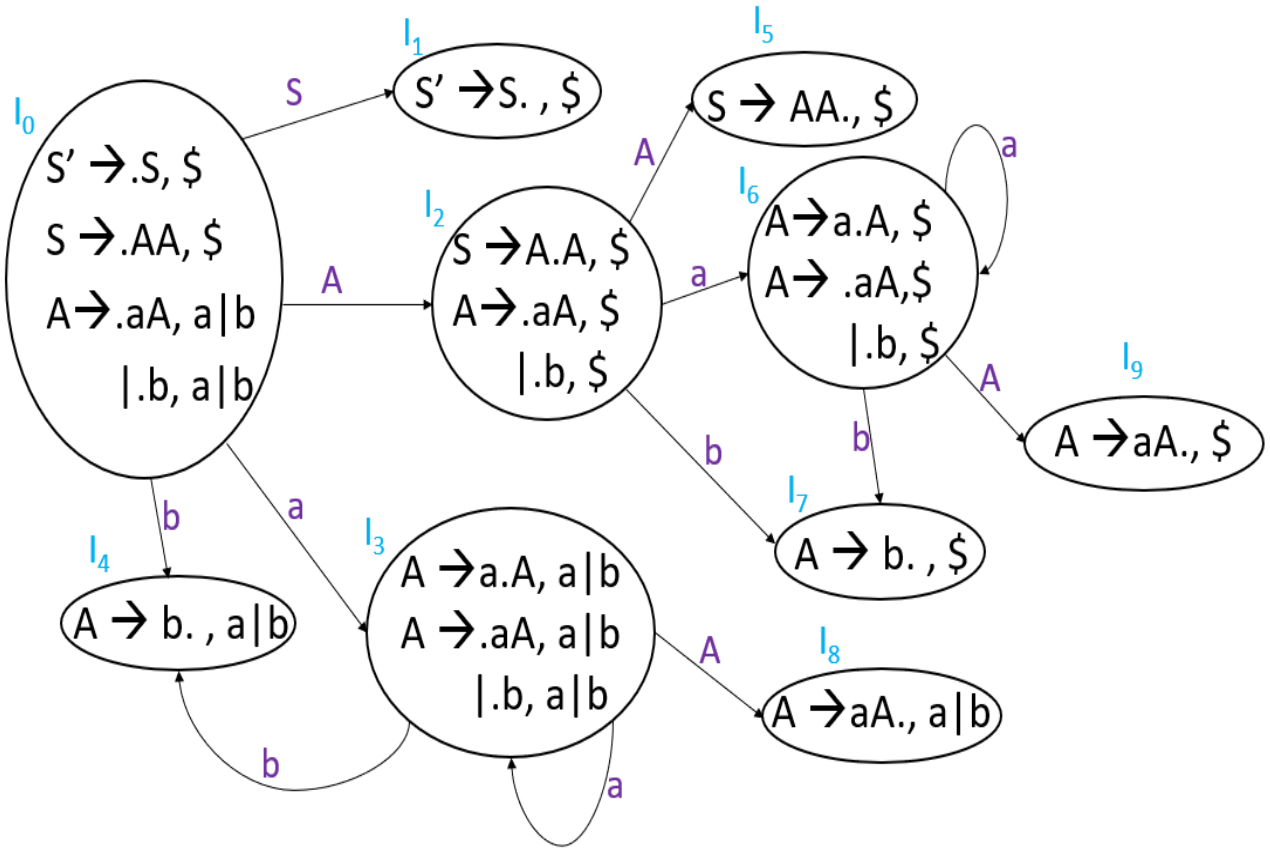
1: S →AA
2: A→aA
3: A → b



- Shift and GOTO moves are same as LR(0) and SLR(1) parsing table. Only difference will be in reduce move
- Size of the parsing table is bigger

CLR(1) Parsing Table					
	Action			Goto	
	a	b	$	S	A
0	s_3	s_4			2
1					
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		

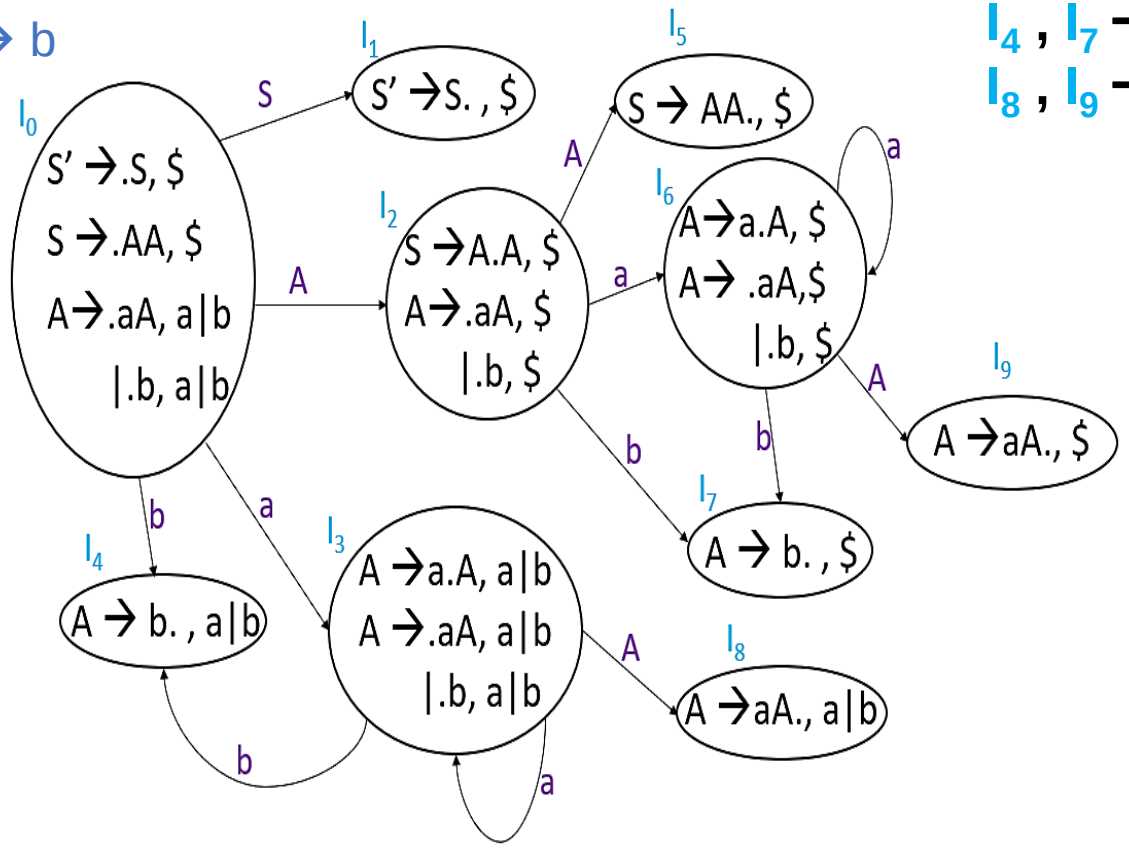
LALR(1) Parsing Table



- In LR(1) items, the numbers of states has increased compared with LR(0) items
- In terms of LR(0) items, both I_3 and I_6 are **same**
- Also, both I_4 and I_7 are **same** in LR(0)
- Similarly, both I_8 and I_9 are **same** in LR(0)
- In LALR(1) parsing table, we **merge** these states
- I_3 , I_6 → I_{36}
- I_4 , I_7 → I_{47}
- I_8 , I_9 → I_{89}

1: S →AA
2: A→aA
3: A → b

I_3 , I_6 → I_{36}
I_4 , I_7 → I_{47}
I_8 , I_9 → I_{89}



- Wherever there is 3 or 6 in the states, we are converting it to 36
- Same will happen for 47 and 89
- By merging we are making the **lookaheads irrelevant**
- So, number of state in LALR(1) is same as LR(0) and SLR(1)

LALR(1) Parsing Table

	Action			Goto	
	a	b	$	S	A
0	S_{36}	S_{47}			2
1					
2	S_{36}	S_{47}			5
36	S_{36}	S_{47}			89
47	r_3	r_3			
5			r_1		
36	S_{36}	S_{47}			89
47			r_3		
89	r_2	r_2			
89			r_2		

1: S →AA
2: A→aA
3: A → b

LALR(1) Parsing Table					
	Action			Goto	
	a	b	$	S	A
0	S_{36}	S_{47}			2
1					
2	S_{36}	S_{47}			5
36	S_{36}	S_{47}			89
47	r_3	r_3			
5			r_1		
36	S_{36}	S_{47}			89
47			r_3		
89	r_2	r_2			
89			r_2		

Merge the states

I_3 , I_6 → I_{36}
I_4 , I_7 → I_{47}
I_8 , I_9 → I_{89}

LALR(1) Parsing Table					
	Action			Goto	
	a	b	$	S	A
0	S_{36}	S_{47}			2
1					
2	S_{36}	S_{47}			5
36	S_{36}	S_{47}			89
47	r_3	r_3	r_3		
5			r_1		
89	r_2	r_2	r_2		

- We erase the same rows
- In the LALR parsing table,36 are having the same elements in each row
- For 47 in reduce move there is difference. So we merge them
- For 89, we can also observe that the reduce move is different. We merge the state 47 accordingly

All the Parsing Tables for the Grammar

S → AA
A → aA | b

LR(0) Parsing Table

	Action			Goto	
	a	b	$	A	S
0	s_3	s_4		2	1
1			Accept		
2	s_3	s_4		5	
3	s_3	s_4		6	
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

SLR(1) Parsing Table

	Action			Goto	
	a	b	$	A	S
0	s_3	s_4		2	1
1			Accept		
2	s_3	s_4		5	
3	s_3	s_4		6	
4	r_3	r_3	r_3		
5			r_1		
6	r_2	r_2	r_2		

CLR(1) Parsing Table

	Action			Goto	
	a	b	$	S	A
0	s_3	s_4			2
1					
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		

LALR(1) Parsing Table

	Action			Goto	
	a	b	$	S	A
0	s_{36}	s_{47}			2
1					
2	s_{36}	s_{47}			5
36	s_{36}	s_{47}			89
47	r_3	r_3	r_3		
5			r_1		
89	r_2	r_2	r_2		

- Whenever the number of reduce move reduces, the conflict also become less
- As the blank spaces increase, the error detection capabilities also increases