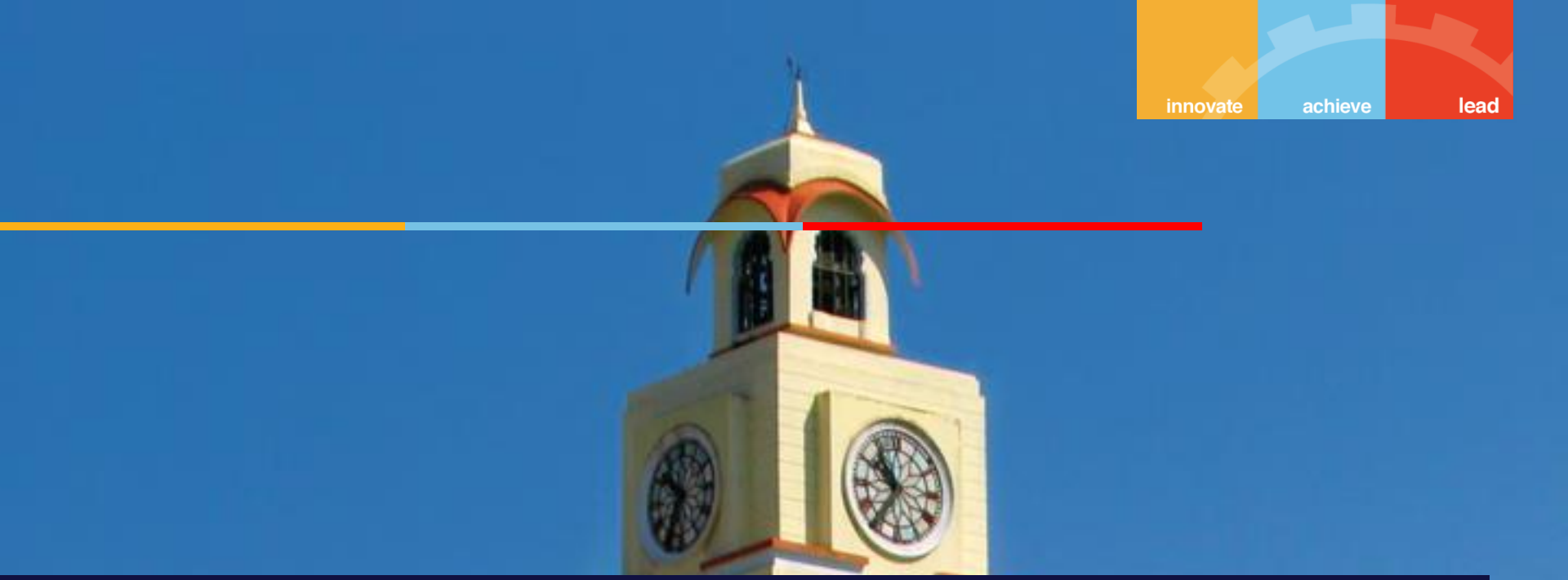


innovate achieve lead

LR(1) Parsers

Dr. Shashank Gupta
Assistant Professor
Department of Computer Science and Information Systems

BITS Pilani
Pilani Campus

Algorithm for CLR (1) Parsing Table

- Construct C={I_0, …, I_n } the sets of LR(1) items.
- If [A → α.aβ, b] is in I_i and goto(I_i , a)=I_j then action[i,a]=shift j
- If [A → α., a] is in I_i then action[i,a] = reduce A → α
- If [S′ → S., $] is in I_i then action[i,$] = accept
- If goto(I_i , A) = I_j then goto[i,A] = j for all non terminals A

More Examples

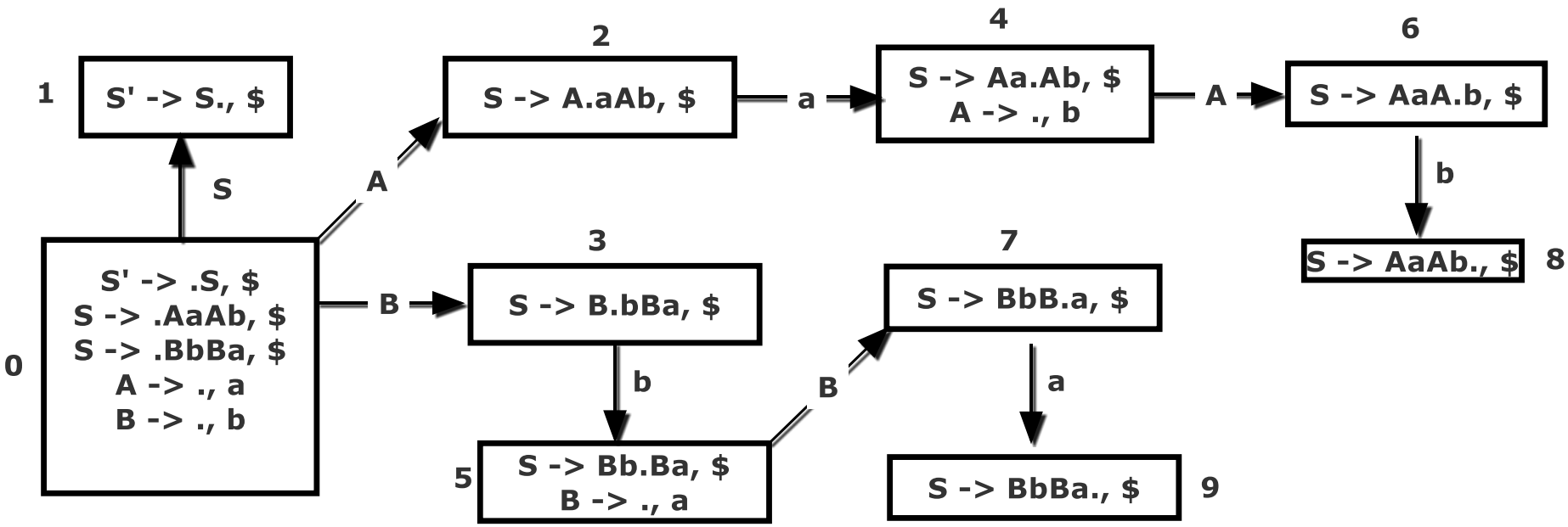
Construct the LR(1) collection of items for the following Grammar and design the CLR(1) parsing table and find out whether this grammar is CLR(1) or not.

```
S -> AaAb | BbBa

A -> ε

B -> ε
```

GOTO GRAPH



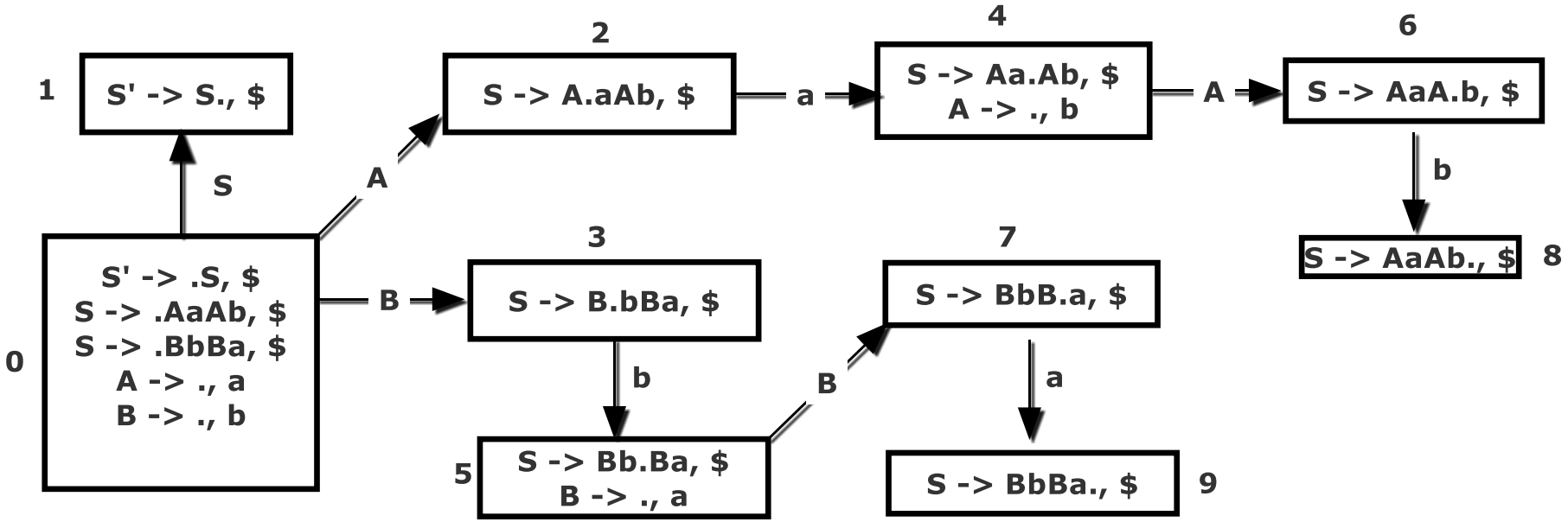
0S' -> S
1S -> AaAb
2S -> BbBa
3A -> ε
4B -> ε

CLR(1) Parsing Table

0S′ -> S
1S -> AaAb
2S -> BbBa
3A -> ε
4B -> ε

BLANK CELLS ARE ERROR ENTRIES

	ACTION			GOTO		
	a	**b**	**$**	**S**	**A**	**B**
0	R3	R4		1	2	3
1			ACCEPT			
2	S4					
3		S5				
4		R3			6	
5	R4					7
6		S8				
7	S9					
8			R1			
9			R2			



CLR(1) Parser

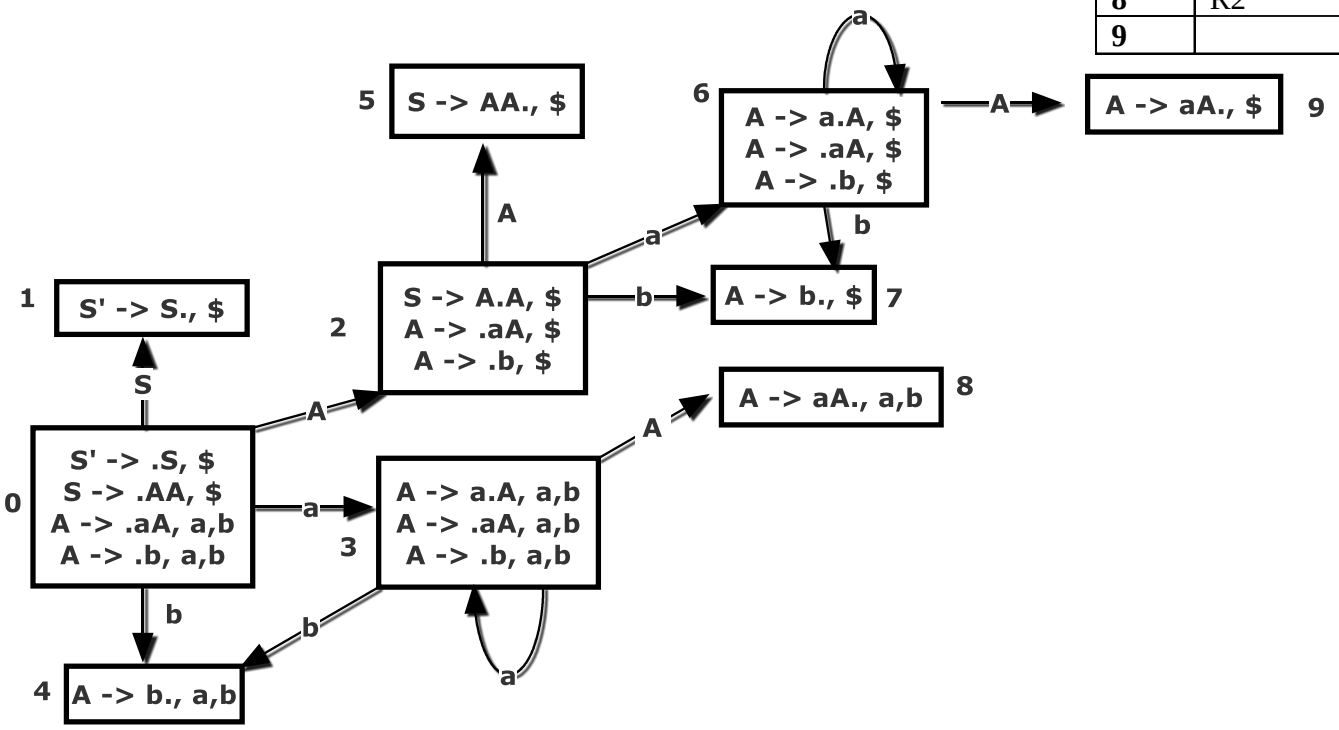
On an error canonical LR parser never makes a wrong shift/reduce move. It immediately declares an error.

Problem: Canonical LR parse table has a large number of states

Revisiting CLR (1) Parsing Table

	ACTION			GOTO	
	a	**b**	**$**	**S**	**A**
0	S3	S4		1	2
1			accept		
2	S6	S7			5
3	S3	S4			8
4	R3	R3			
5			R1		
6	S6	S7			9
7			R3		
8	R2	R2			
9			R2		

BLANK CELLS ARE ERROR ENTRIES



0S’ -> S
1S -> AA
2A -> aA
3A -> b

LookAhead (LALR(1) Parser)

Consider a pair of similar looking states (same core and different lookaheads) in the set of LR(1) items

- I_4 : A → b. , a/b I_7 : A → b., $

Replace I_4 and I_7 by a new state I_{47} consisting of (A → b., a/b/$)

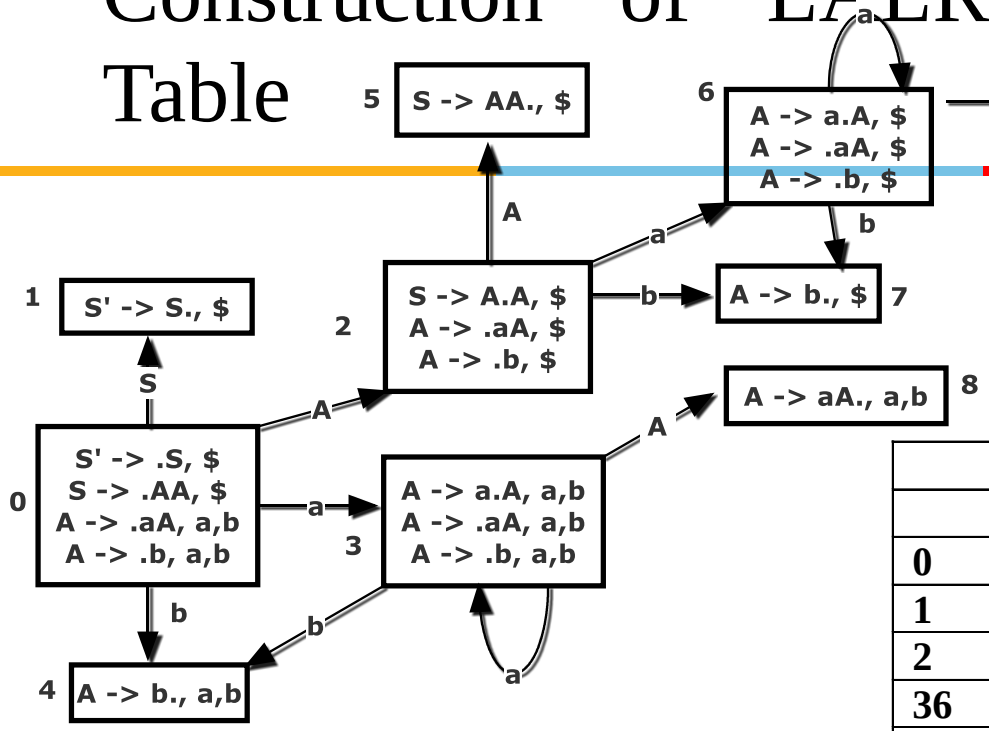
- Similarly I_3 & I_6 and I_8 & I_9 form pairs.
- Merge LR(1) items having the same core.

Construction of LALR(1) Parse Table

Construct $C=\{I_0, \ldots\ldots, I_n\}$ set of LR(1) items

For each LR(0) item present in LR(1) items find all sets having the same LR(0) item and replace these sets by their union.

Construction of LALR(1) Parse Table



	ACTION			GOTO	
	a	**b**	**$**	**S**	**A**
0	S3	S4		1	2
1			accept		
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(1) PARSE TABLE

	ACTION			GOTO	
	a	**b**	**$**	**S**	**A**
0	S36	S47		1	2
1			accept		
2	S36	S47			5
36	S36	S47			89
47	R3	R3	R3		
5			R1		
89	R2	R2	R2		

Construction of LALR(1) Parse Table

Let C' = {J_0 ,……., J_m} be the resulting set of items.

- Construct action table as was done earlier.

Let J = I_1 U I_2…….U I_k since I_1 , I_2……., I_k have same core, goto(J,X) will have the same core.

- Let K = goto(I_1,X) U goto(I_2,X)……goto(I_k,X) the goto(J,X)=K

LALR(1) Parse Table

In general core is a set of LR(0) items and LR(1) grammar may produce more than one set of items with the same core.

Merging items in LALR(1) parsing table may produces conflicts.

- SLR and LALR parse tables have same number of states

LALR(1) Parse Table

Merging items may result into conflicts in LALR parsers which did not exist in LR parsers

New conflicts may arise in LALR(1) Parsers.

LALR(1) Parse Table

LALR parser can have new conflicts

- Assume states {[X→α., a], [Y→β., b]} and {[X→α., b], [Y→β., a]}
- Merging the two states produces

{[X→α., a/b], [Y→β., a/b]}

More Examples

Consider the following Grammar and find out whether it is LALR(1) or not.

S -> aEa | bEb | aFb | bFa

E -> e

F -> e

GRAMMAR IS NOT LALR(1) BECAUSE OF CONFLICT IN SOME OF ITS STATES.