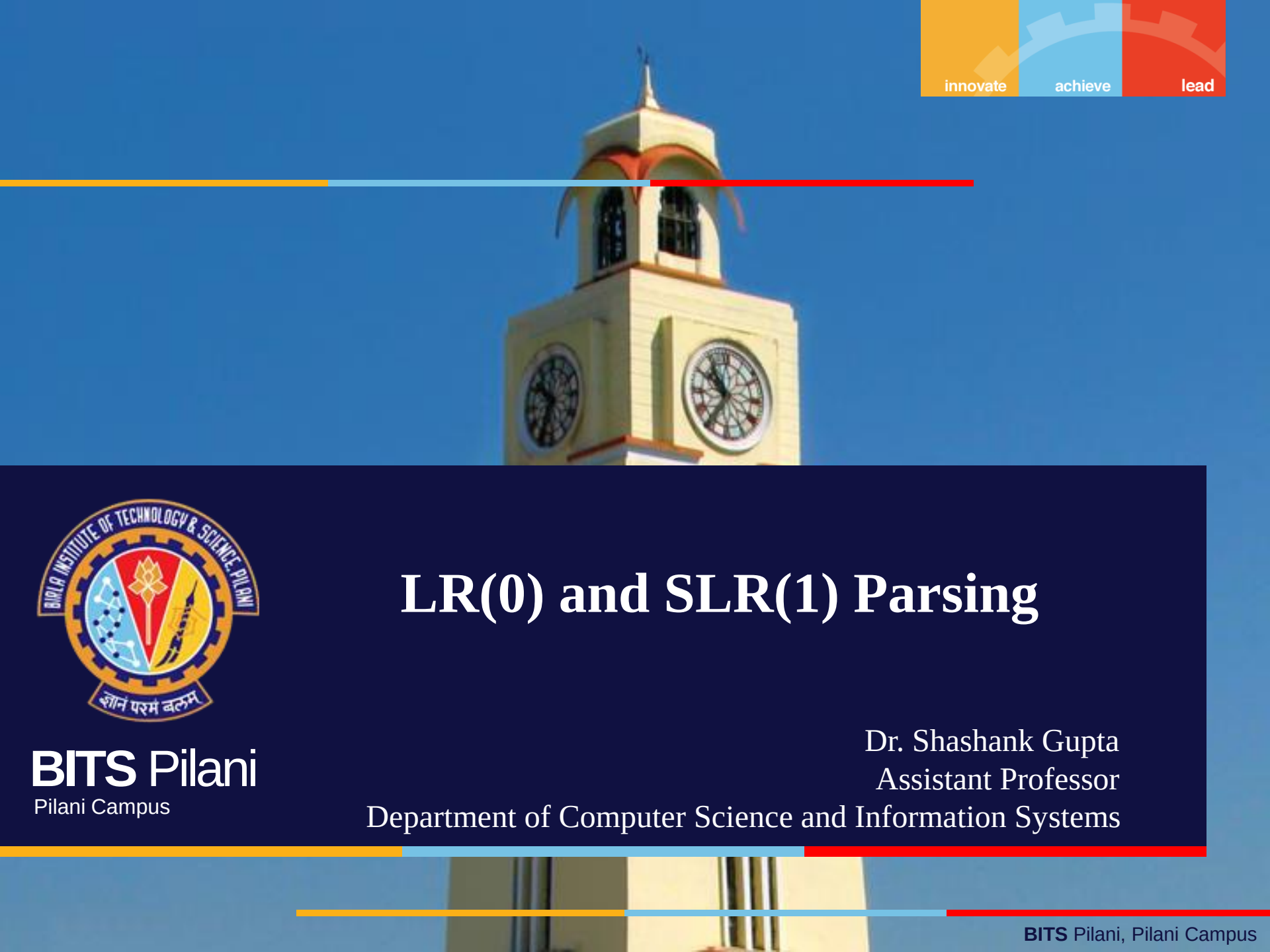


innovate achieve lead

LR(0) and SLR(1) Parsing

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

BITS Pilani
Pilani Campus

LR(0) Parser Example

Construct a LR (0) parsing table for the following grammar

$$S \rightarrow A\,A$$

$$A \rightarrow a\,A \mid b$$

In addition, parse the following i/p: aabb using LR (0) parsing table.

Augment the Grammar

$$S \rightarrow A\,A$$

$$A \rightarrow a\,A \mid b$$

$$0: S' \rightarrow S$$

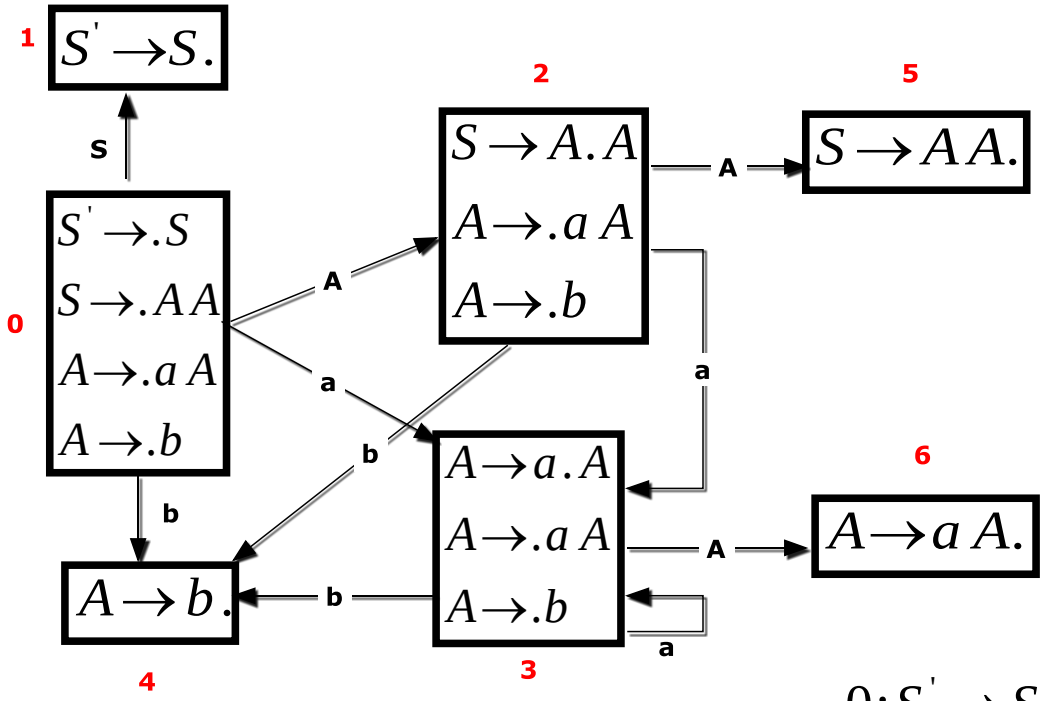
$$1: S \rightarrow A\,A$$

$$2: A \rightarrow a\,A$$

$$3: A \rightarrow b$$

Goto Graph and Parsing Table

GOTO GRAPH



LR(0) PARSING TABLE

	Action			Goto	
	a	**b**	**$**	**S**	**A**
0	S3	S4		1	2
1			accept		
2	S3	S4			5
3	S3	S4			6
4	R3	R3	R3		
5	R1	R1	R1		
6	R2	R2	R2		

$0: S' \rightarrow S$

$1: S \rightarrow A\,A$

$2: A \rightarrow a\,A$

$3: A \rightarrow b$

BLANK CELLS ARE ERROR ENTRIES

S: SHIFT

R:REDUCE

LR(0) Parsing Example

Stack	Input	Action
0	a a b b $	**Shift:** Push a and state 3 on stack and increment ip++.
0a3	abb$	**Shift**: Push a and state 3 on stack and increment ip++.
0a3a3	bb$	**Shift**: Push b and state 4 on stack and increment ip++.
0a3a3b4	b$	**Reduce** by A->b. Pop 2 symbols from the stack and push A and state 6 on to the stack.
0a3a3A6	b$	**Reduce** by A->aA. Pop 4 symbols from the stack and push A and state 6 on to the stack.

	Action			Goto	
	a	**b**	**$**	**S**	**A**
0	S3	S4		1	2
1			accept		
2	S3	S4			5
3	S3	S4			6
4	R3	R3	R3		
5	R1	R1	R1		
6	R2	R2	R2		

$0: S' \rightarrow S$

$1: S \rightarrow A\,A$

$2: A \rightarrow a\,A$

$3: A \rightarrow b$

LR(0) Example Parsing

Stack	Input	Action
0a3a3A6	b$	**Reduce** by A->aA. Pop 4 symbols from the stack and push A and state 6 on to the stack.
0a3A6	b$	**Reduce** by A->aA. Pop 4 symbols from the stack and push A and state 2 on to the stack.
0A2	b$	**Shift:** Push b and state 4 on stack and increment ip++.
0A2b4	$	**Reduce** by A->b. Pop 2 symbols from the stack and push A and state 5 on to the stack.
0A2A5	$	**Reduce** by S->AA. Pop 4 symbols from the stack and push S and state 1 on to the stack.
0S1	$	**Accept**

	Action			Goto	
	a	**b**	**$**	**S**	**A**
0	S3	S4		1	2
1			accept		
2	S3	S4			5
3	S3	S4			6
4	R3	R3	R3		
5	R1	R1	R1		
6	R2	R2	R2		

$0: S' \rightarrow S$

$1: S \rightarrow A\,A$

$2: A \rightarrow a\,A$

$3: A \rightarrow b$

LR(0) Grammar

A Grammar is LR(0) if its LR(0) parsing table does not contain multiple defined entries

- OR

A Grammar is LR(0) if it does not have any shift-reduce/reduce-reduce conflicts in any of its states.

Possible Actions in LR Parser

Assume S_i is on top of stack and a_i is current input symbol.

Action $[S_i, a_i]$ can have four values.

- **sj**: shift a_i to the stack, goto state j.
- **rk**: reduce by production rule number k
- **Accept**
- **Error**

Contents of LR Parser

- The following tuple defines a configuration of a LR parser

  <Stack Contents, Remaining Input>

- Initially the configuration is

  <S_0, a_0, a_1, ----, a_n $ >

- Typical final configuration on a successful parse is

  < S_0X_1 S_i , $>

Execution of LR (0) Parser

Stack: $S_0X_1\ S_1X_2 \ldots X_mS_m$ Input: $a_i\ a_{i+1} \ldots a_n\$$

- If action$[S_m,a_i\ ]$ = shift S
  Then the configuration becomes
  Stack: $S_0X_1\ S_1X_2 \ldots X_mS_m\ a_i\ S$ Input: $a_{i+1} \ldots a_n\$$

- If action$[S_m,a_i\ ]$ = reduce $A \rightarrow \beta$
  Then the configuration becomes
  Stack: $S_0X_1\ S_1 \ldots X_{m-r}\ S_{m-r}\ AS$ Input: $a_i\ a_{i+1} \ldots a_n\$$
  where $r = |\beta|$ and $S = goto[S_{m-r}, A]$

Execution of LR (0) Parser

Stack: $S_0X_1\ S_1X_2 \ldots X_mS_m$ Input: $a_i\ a_{i+1} \ldots a_n\$$

- If action$[S_m,a_i\ ]$ = accept
  Then parsing is completed. HALT
- If action$[S_m,a_i\ ]$ = error (or empty cell)
  Then invoke error recovery routine.

LR Parsing Algorithm

```
Initial state: Stack: S0 Input: w$
while (1) {
      if (action[S,a] = shift S') {
            push(a); push(S'); ip++
      } else if (action[S,a] = reduce A → β) {
            pop (2*|β|) symbols;
            push(A); push (goto [S'',A])
      (S'' is the state at top of the stack after
popping the symbols)
      } else if (action[S,a] = accept) {
            exit
      } else { error }
```

Example

- Construct a LR (0) parsing table on the following grammar

$$S \rightarrow (L)$$

$$S \rightarrow x$$

$$L \rightarrow S$$

$$L \rightarrow L, S$$

Parse the following input: **(x,(x))**

Example

Consider the following Grammar and find out whether it is LR (0) or not.

$$E \rightarrow T + E$$

$$E \rightarrow T$$

$$T \rightarrow id$$

Augmented Grammar

$$E' \rightarrow E$$
$$E \rightarrow T + E$$
$$E \rightarrow T$$
$$E \rightarrow id$$

$$E \rightarrow T + E$$
$$E \rightarrow T$$
$$T \rightarrow id$$

Example

The Grammar is not LR(0) since, its parse table has multiple defined entries in the form of Shift-Reduce Conflict.

$0\ E' \rightarrow E$

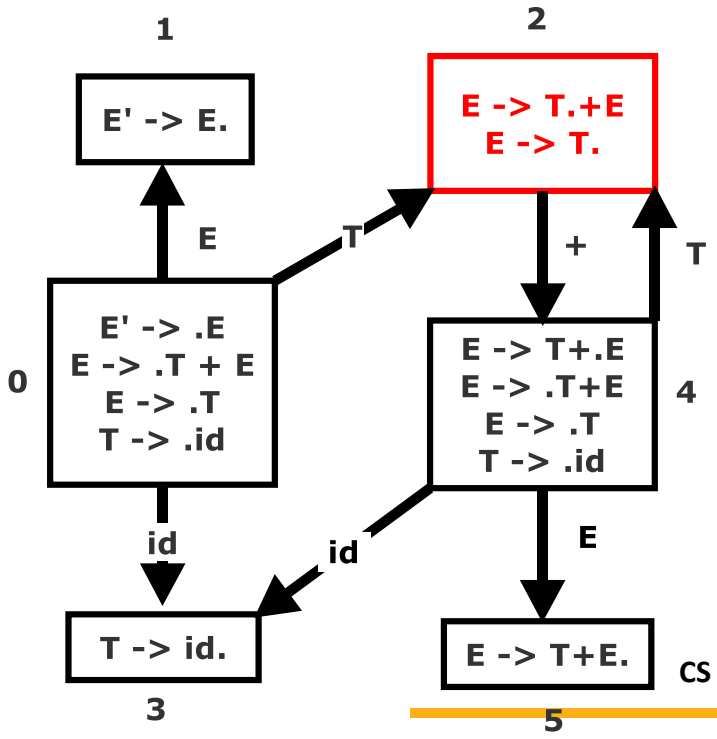
$1\ E \rightarrow T + E$

$2\ E \rightarrow T$

$3\ E \rightarrow id$

	ACTION			GOTO	
	+	id	$	E	T
0		S3		1	2
1			ACCEPT		
2	S4/R2	R2	R2		
3	R3	R3	R3		
4		S3		5	2
5	R1	R1	R1		

LR(0) PARSING TABLE



$E' \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

$E \rightarrow id$

Simple LR (SLR) Parsing

The SLR (1) parser is similar to LR(0) parser except that the reduced entry.

- LR(0) parser always focusses on blind reductions.

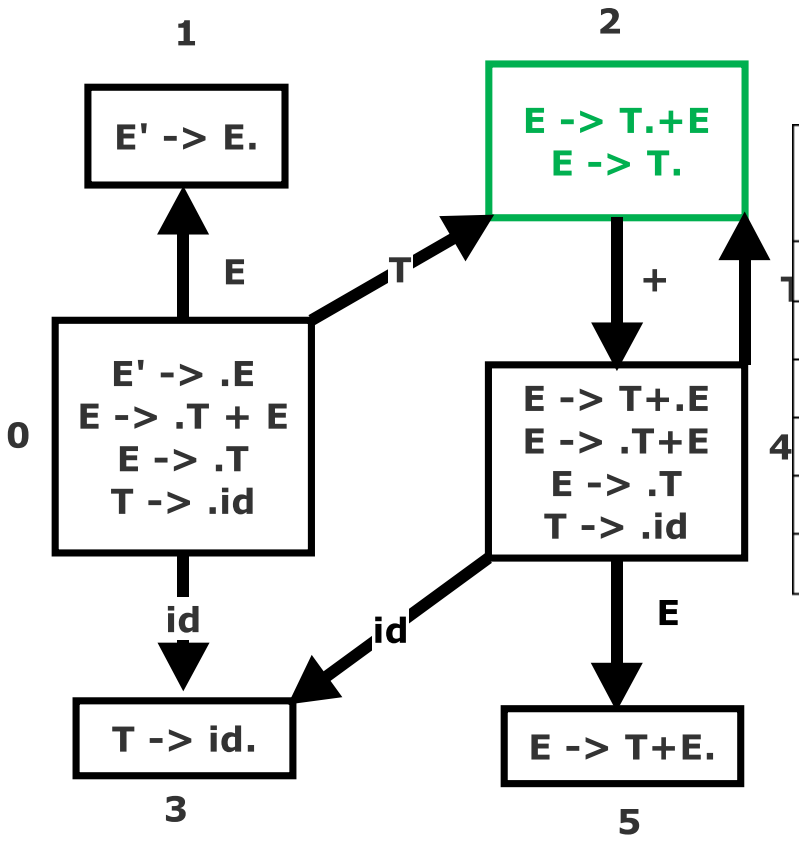
The reduced productions are written only in the FOLLOW of the variable whose production is reduced.

Algorithm for SLR(1) Parsing Table

- Construct $C=\{I_0, \ldots, I_n\}$ the collection of sets of LR(0) items
- If $A \rightarrow \alpha.a\beta$ is in I_i and $goto(I_i,a) = I_j$ then action[i,a] = shift j
- If $A \rightarrow \alpha.$ is in I_i then action[i,a] = reduce $A \rightarrow \alpha$ for all a in **follow(A)**
- If $S' \rightarrow S.$ is in I_i then action[i,$] = accept
- If $goto(I_i, A) = Ij$ then goto[i,A]=j for all non terminals A
- All entries which are not defined are errors

SLR (1) Parsing

BLANK CELLS ARE ERROR ENTRIES



SLR (1) PARSING TABLE

	ACTION			GOTO	
	+	id	$	E	T
0		S3		1	2
1			ACCEPT		
2	S4		R2		
3	R3		R3		
4		S3		5	2
5			R1		

$0\ E^{'} \rightarrow E$

$1\ E \rightarrow T + E$

$2\ E \rightarrow T$

$3\ E \rightarrow id$

$E^{'} \rightarrow E$

$E \rightarrow T + E$

$E \rightarrow T$

$E \rightarrow id$