LALR(1)

Dr. Meera Thapar Khanna

CSE Department

Pandit Deendayal Energy University

LALR(1) Parser

- An LALR(1) parser is an LR(1) parser in which all states that differ only in the look-ahead components of the configurations are merged and union of look-aheads.
- LALR (Look Ahead LR) Parser reduces the number of states to the same as SLR(1) Parser.
- LALR (Look Ahead LR) Parser still retains some of the power of the LR(1) look-aheads.

Construction of LALR(1) Parser

1. Construct all canonical LR(1) states.
2. Merge those states that are identical if the lookaheads are ignored, i.e., two states being merged must have the same number of items and the items have the same core.
3. The Goto Function for the new LALR(1) state is the union of the merged states.
4. The action and goto entries are constructed from the LALR(1) states as for the canonical LR(1) parser.

Example

Grammar

$S \rightarrow CC$

$C \rightarrow cC$

$C \rightarrow d$

Step1: Augmented Grammar

Augmented Grammar:

S’ →•S$	0
S → •CC	1
C → •cC	2
C → •d	3

Step2: Construct LR(1) closure items

I0:

S’ → •S, $	0	(Look-ahead of 1st production is always $)
S → •CC, $	1	(nothing after S in 0th production so look-ahead for 1st prodution is same $)
C → •cC, c|d	2	(C is there after C so look-ahead for 2nd production is First(C) i.e. {c,d})
C → •d, c|d	3	(C is there after c so look-ahead for 3rd production is First(C) i.e. {c,d})

- Goto(i0, S) = S’ → S•, $ = i1

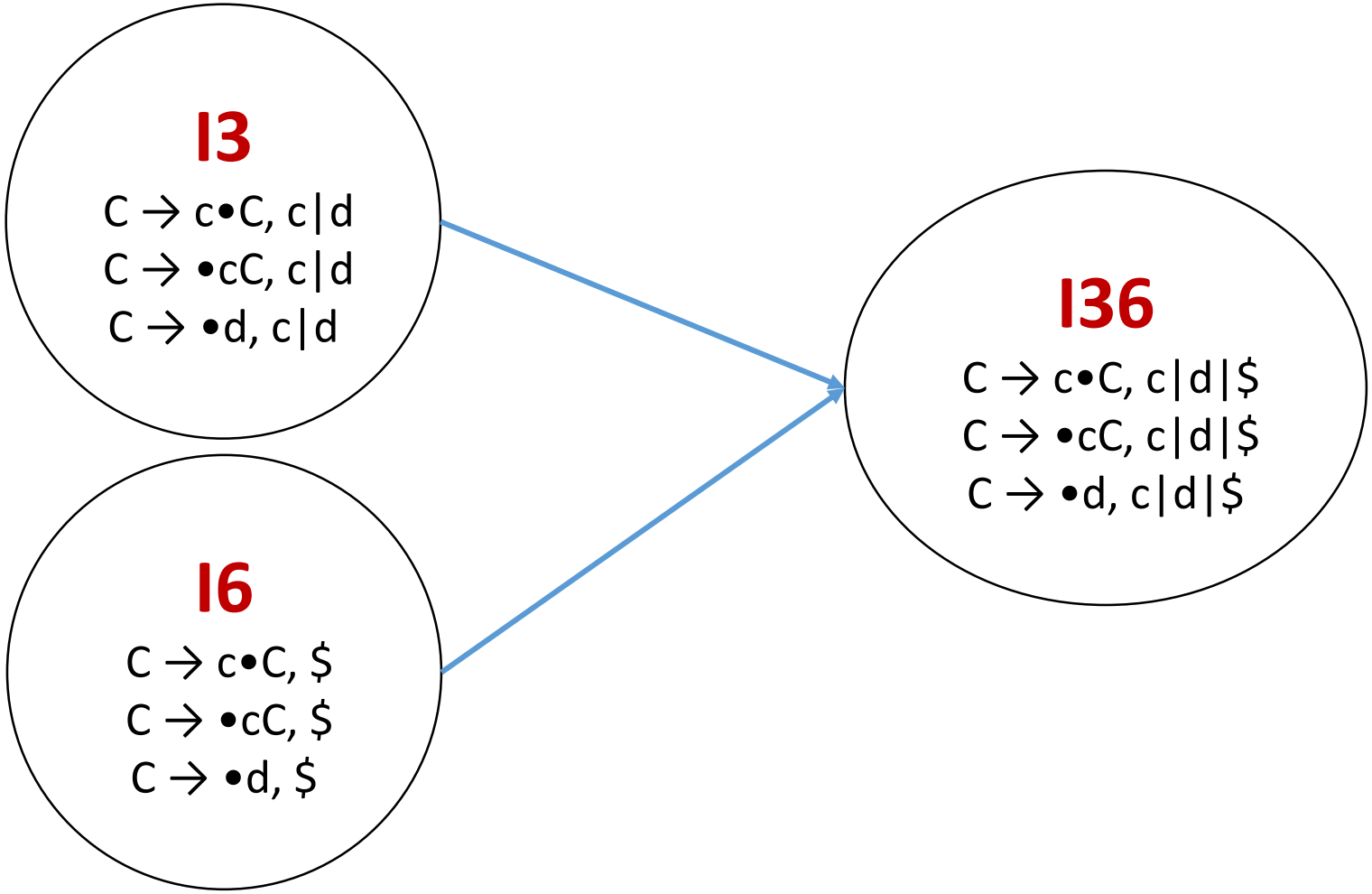
- Goto(i0, C) = S → C•C, $ = i2
  C → •cC, $
  C → •d, $

- Goto(i0, c) = C → c•C, c|d = i3
  C → •cC, c|d
  C → •d, c|d

- Goto(i0, d) = C → d•, c|d = i4

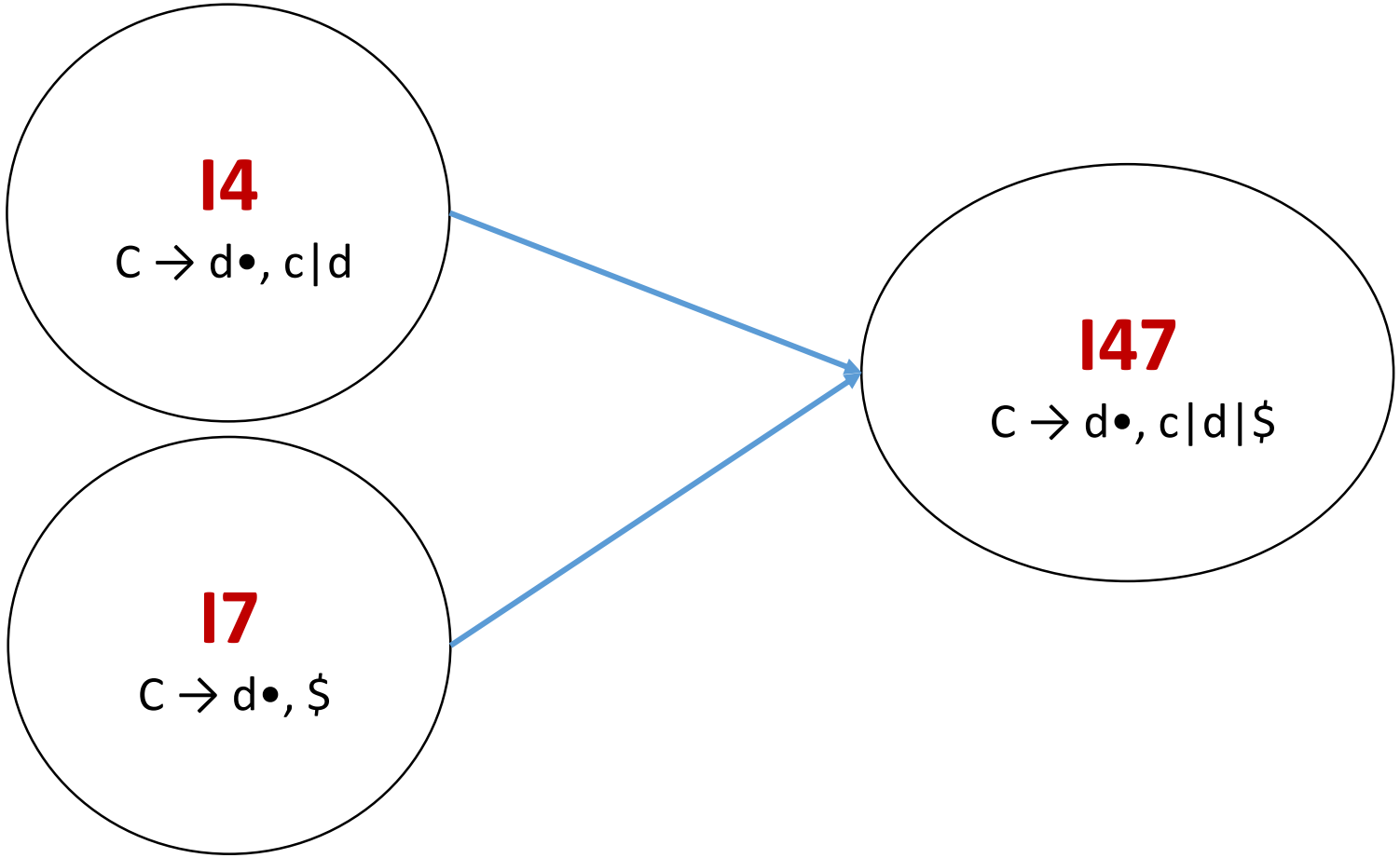
- Goto(i2, C) = C → CC•, $ = i5
- Goto(i2, c) = C → c•C, $ = i6
  C → •cC, $
  C → •d, $
- Goto(i2, d) = C → d•, c|d = i7
- Goto(i3, C) = C → cC•, c|d = i8
- Goto(i3, c) = C → c•C, c|d = i3
  C → •cC, c|d
  C → •d, c|d
- Goto(i3, d) = C → d•, c|d = i4
- Goto(i6, C) = C → cC•, $ = i9
- Goto(i6, c) = C → c•C, $ = i6
  C → •cC, $
  C → •d, $
- Goto(i6, d) = C → d•, $ = i7

Identify common LR(1) items and merge



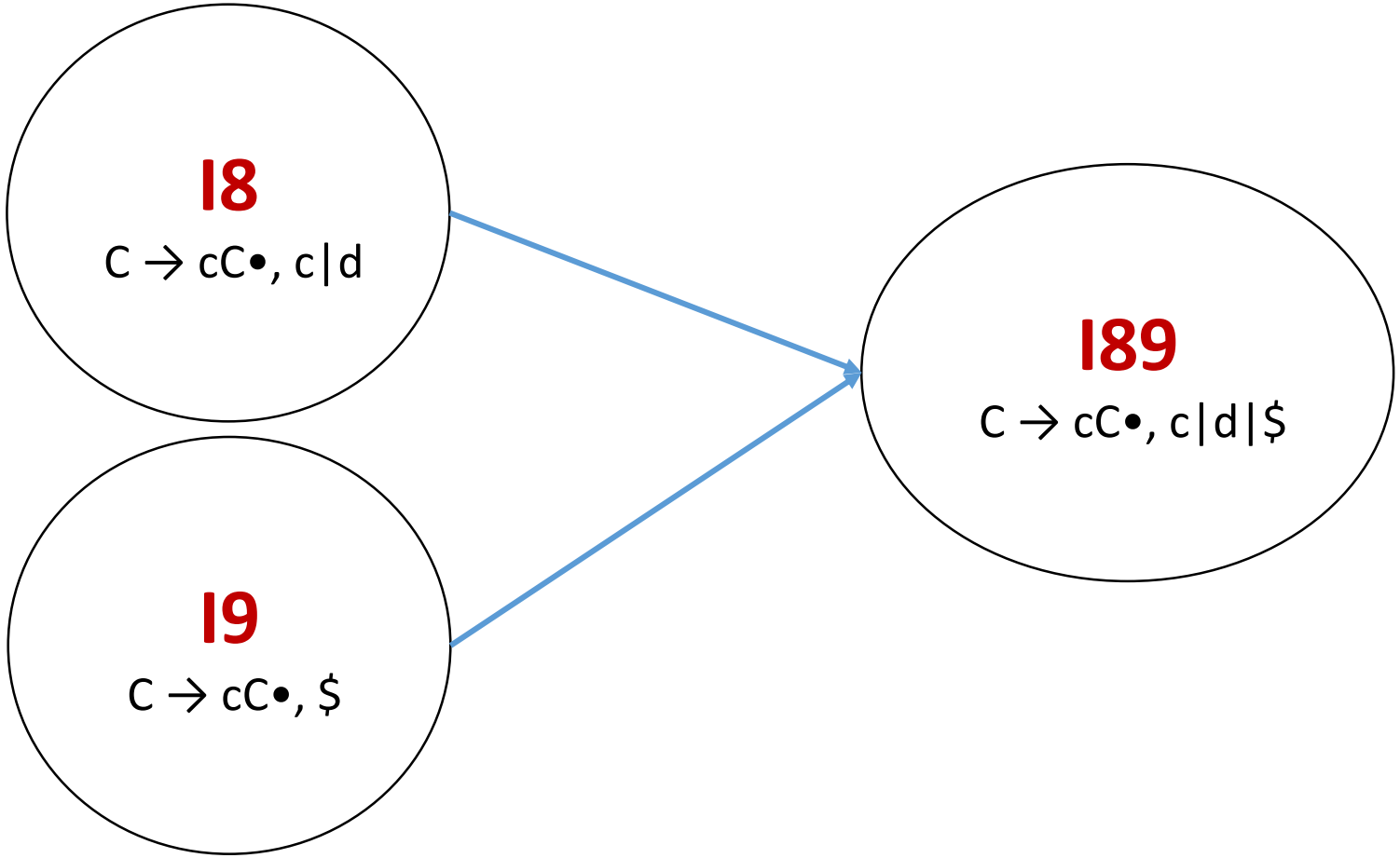
These states are differing only in the look-aheads. They have the same productions. Hence these states are combined to form a single state called as I36

Identify common LR(1) items and merge



These states are differing only in the look-aheads. They have the same productions. Hence these states are combined to form a single state called as I47

Identify common LR(1) items and merge



These states are differing only in the look-aheads. They have the same productions. Hence these states are combined to form a single state called as I89

LALR(1) Parsing Table

states	c	d	$	S	C
	Action Part			GOTO Part	
I0	s36	s47		1	2
I1			acc		
I2	s36	s47			5
I36	s36	s47			89
I47	r3	r3	r3		5
I5			r1		
I89	r2	r2	r2		

Conflicts in LALR(1) Parsers

- A shift-reduce conflict cannot exist in a merged set unless the conflict was already present in one of the original LR(1) configuration sets. When merging, the two sets must have the same core items. If the merged set has a configuration that shifts on ‘a’ and another that reduces on ‘a’, both configurations must have been present in the original sets, and at least one of those sets had a conflict already.

- There could be an reduce-reduce conflict.

  Example:

  I1: A -> d., a
  B -> d., c

  I5: A -> d., c
  B -> d., a

  When merged I15= A-> d., a|c
  B -> d., c|a