UCS1602 - COMPILER DESIGN

Lookahead Canonical LR Parser- (LALR)

Session Objectives

- Understanding LALR parser

Session Outcomes

- At the end of this session, participants will be able to
  - Construct LALR parser
  - Parse using LALR parser

Agenda

- LALR parser
  - LR(1) items
  - Table construction
  - parsing

Introduction

- If we can merge states with same core of LR(0) items, we get a parsing table of lesser number of states or rows. For some LR(1) grammar this merging will not lead to multiple entries in the parsing table and the corresponding grammar is known as LALR (lookahead LR) grammar.

LALR Parsing Tables

1. **LALR** stands for **Lookahead LR.**
2. LALR parsers are often used in practice because LALR parsing tables are smaller than LR(1) parsing tables.
3. The number of states in SLR and LALR parsing tables for a grammar G are equal.
4. But LALR parsers recognize more grammars than SLR parsers.
5. ***yacc*** creates a LALR parser for the given grammar.
6. A state of LALR parser will be again a set of LR(1) items.

Creating LALR Parsing Tables

Canonical LR(1) Parser → LALR Parser

shrink similar core state

- This shrink process may introduce a **reduce/reduce** conflict in the resulting LALR parser (so the grammar is NOT LALR)
- But, this shrik process does not produce a **shift/reduce** conflict.

The Core of A Set of LR(1) Items

- The core of a set of LR(1) items is the set of its first component.

Ex: $S \rightarrow L \bullet =R,\$$ ➔ $S \rightarrow L \bullet =R$ ← Core

$R \rightarrow L \bullet ,\$$ $R \rightarrow L \bullet$

- We will find the states (sets of LR(1) items) in a canonical LR(1) parser with same cores. Then we will merge them as a single state.

I_1: $L \rightarrow id \bullet ,=$

➔

A new state: I_{12}: $L \rightarrow id \bullet ,=$

$L \rightarrow id \bullet ,\$$

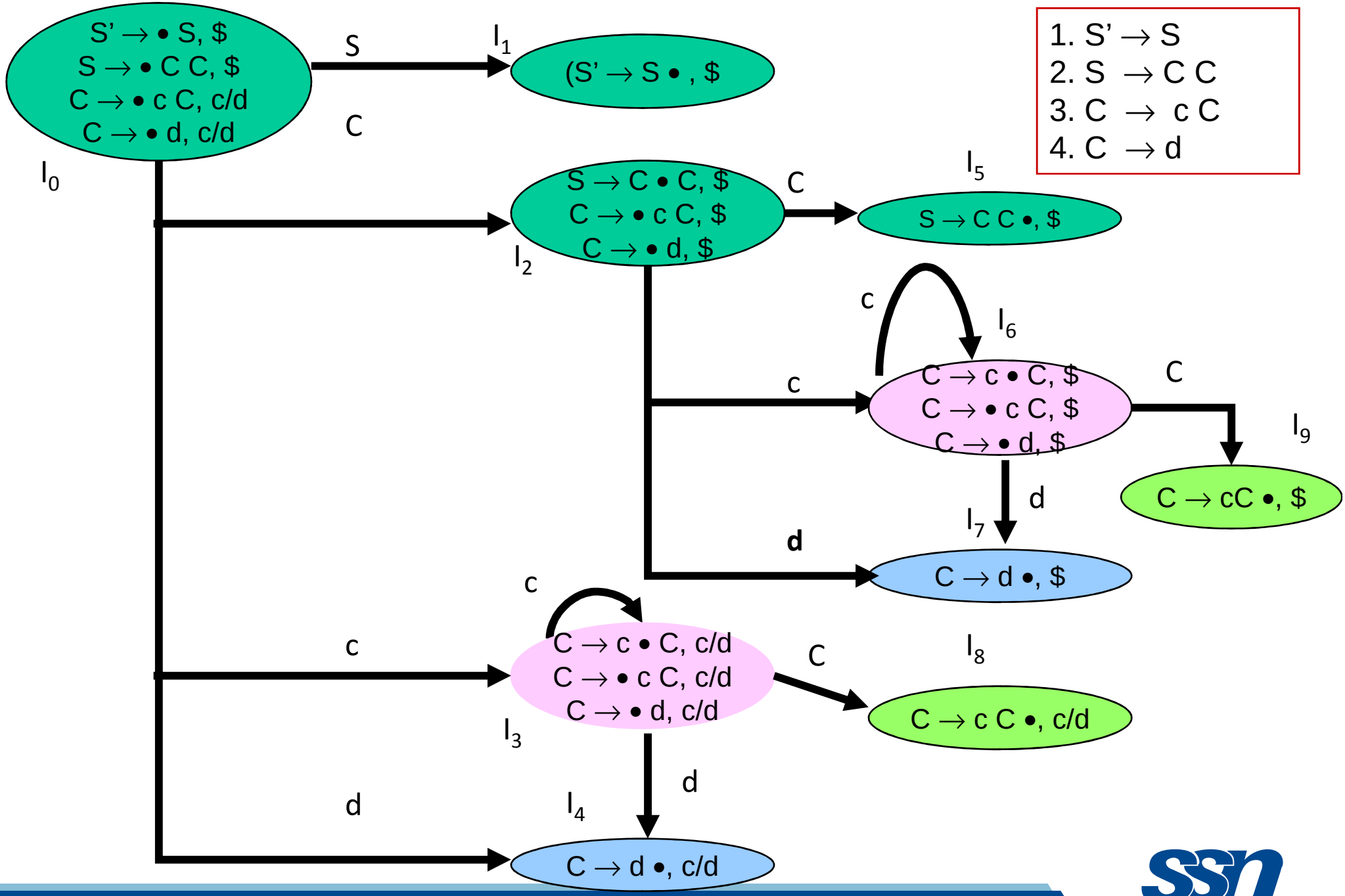
I_2: $L \rightarrow id \bullet ,\$$

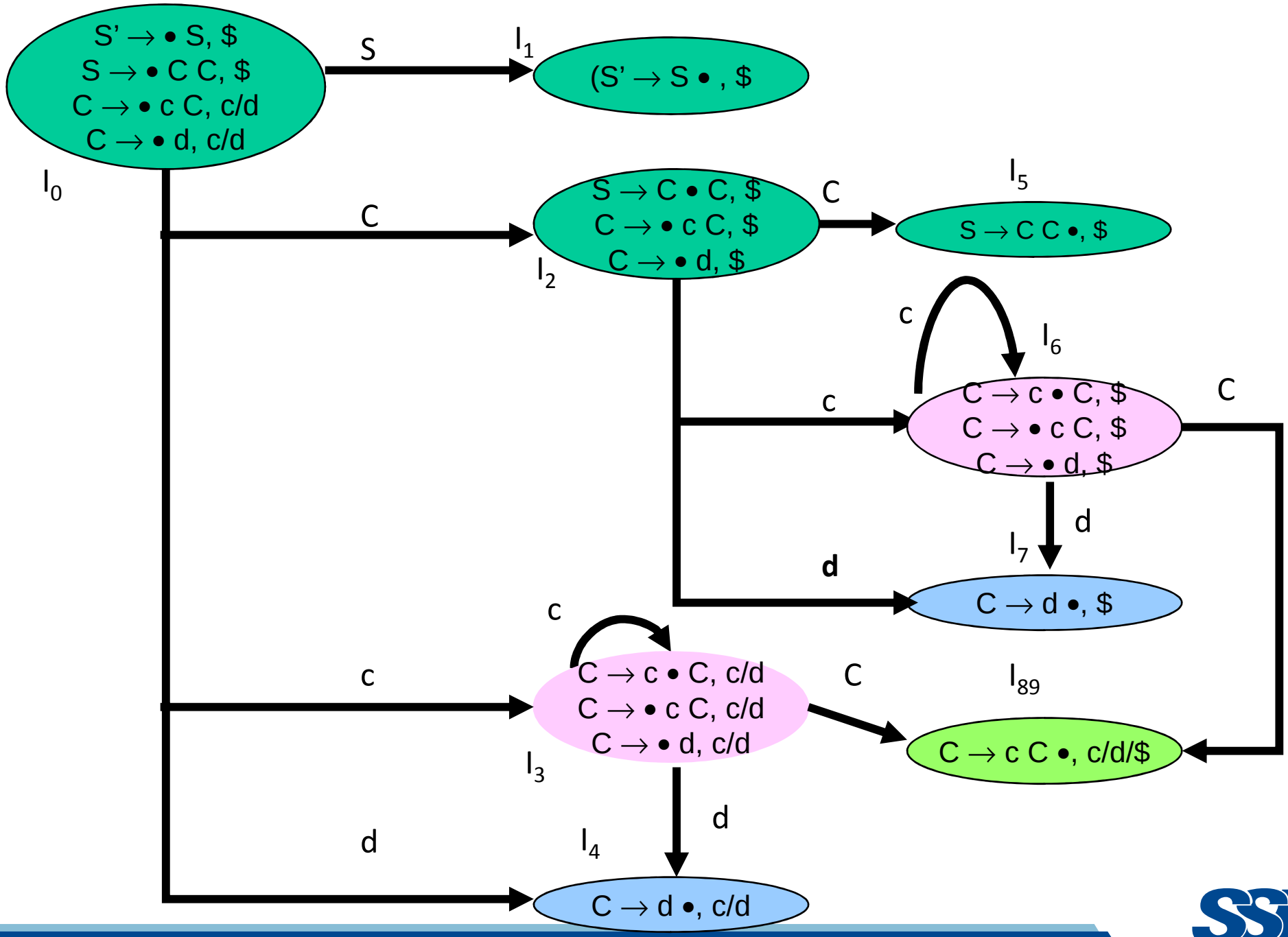
have same core, merge them

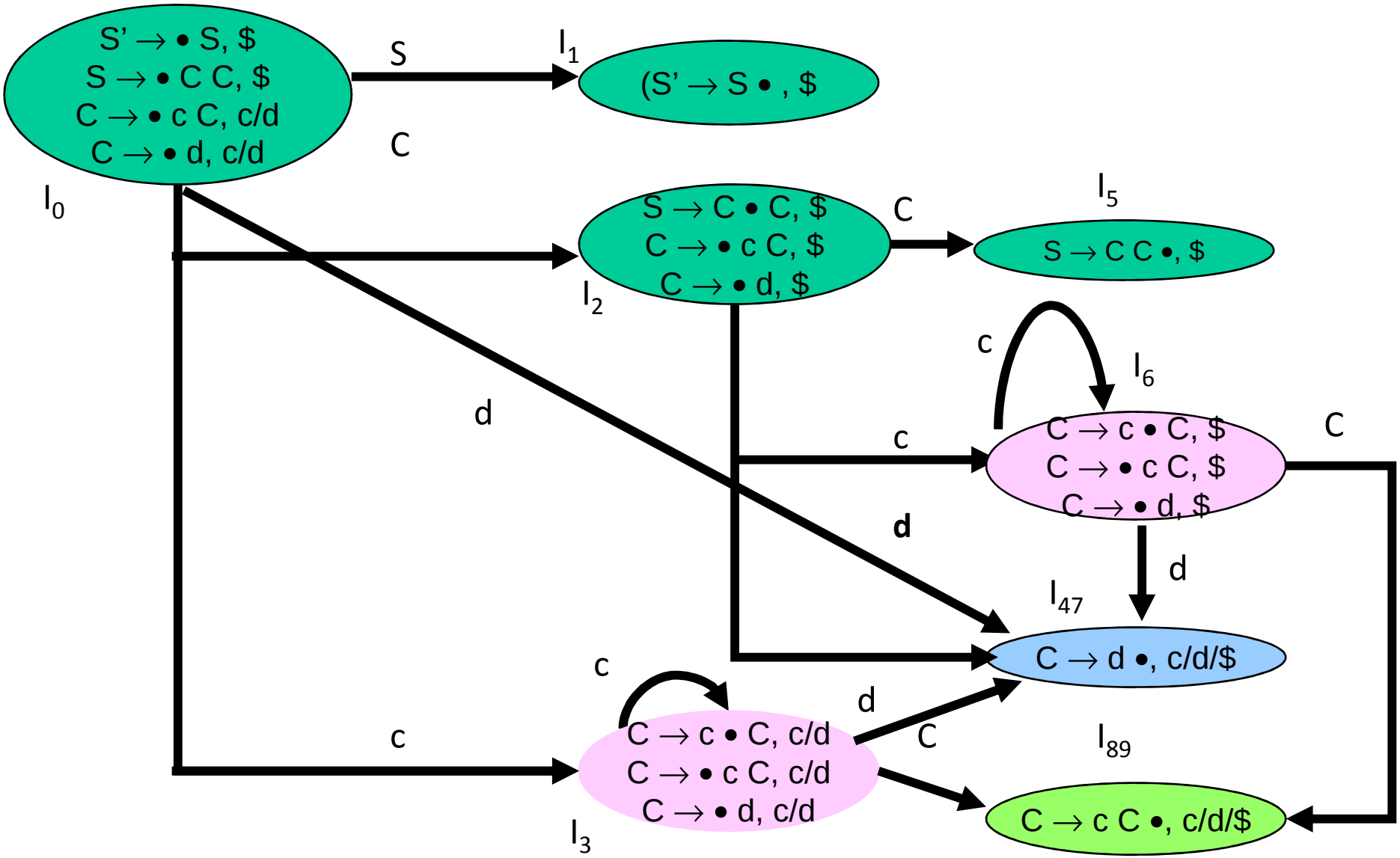
- We will do this for all states of a canonical LR(1) parser to get the states of the LALR parser.
- In fact, the number of the states of the LALR parser for a grammar will be equal to the number of states of the SLR parser for that grammar.

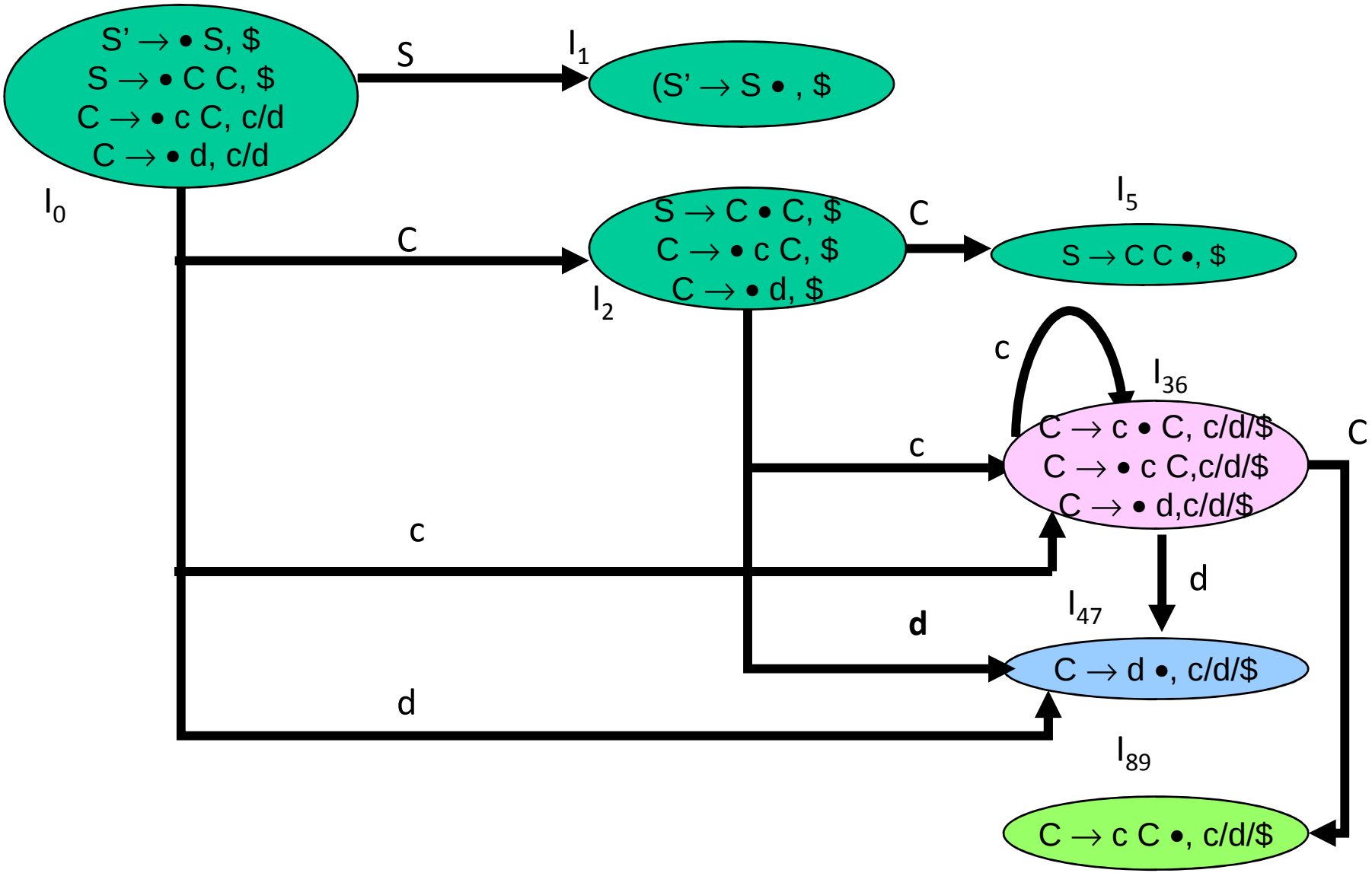
Creation of LALR Parsing Tables

1. Create the canonical LR(1) collection of the sets of LR(1) items for the given grammar.
2. For each core present; find all sets having that same core; replace those sets having same cores with a single set which is their union.
3. Create the parsing tables (action and goto tables) same as the construction of the parsing tables of LR(1) parser.
    1. Note that: If $J=I_1 \cup \ldots \cup I_k$ since $I_1,\ldots,I_k$ have same cores
        ➔ cores of goto(I_1,X),...,goto(I_2,X) must be same.
    1. So, goto(J,X)=K where K is the union of all sets of items having same cores as goto(I_1,X).
4. If no conflict is introduced, the grammar is LALR(1) grammar.


S' → • S, $
S → • C C, $
C → • c C, c/d
C → • d, c/d
I0
S
I1
(S' → S • , $
C
S → C • C, $
C → • c C, $
C → • d, $
I2
C
I5
S → C C •, $
c
I6
C → c • C, $
C → • c C, $
C → • d, $
c
C
I9
C → cC •, $
d
I7
d
C → d •, $
c
C → c • C, c/d
C → • c C, c/d
C → • d, c/d
I3
c
C
I8
C → c C •, c/d
d
I4
d
C → d •, c/d
1. S' → S
2. S → C C
3. C → c C
4. C → d


S' → • S, $
S → • C C, $
C → • c C, c/d
C → • d, c/d
I0
S
I1
(S' → S • , $
C
S → C • C, $
C → • c C, $
C → • d, $
I2
C
I5
S → C C •, $
c
I6
C → c • C, $
C → • c C, $
C → • d, $
c
C
d
I7
d
C → d •, $
c
C → c • C, c/d
C → • c C, c/d
C → • d, c/d
I3
c
C
I89
C → c C •, c/d/$
d
I4
d
C → d •, c/d


S' → • S, $
S → • C C, $
C → • c C, c/d
C → • d, c/d
I0
S
I1
(S' → S • , $
C
S → C • C, $
C → • c C, $
C → • d, $
I2
C
I5
S → C C •, $
c
I6
d
c
C → c • C, $
C → • c C, $
C → • d, $
C
d
d
I47
C → d •, c/d/$
c
c
C → c • C, c/d
C → • c C, c/d
C → • d, c/d
I3
d
C
I89
C → c C •, c/d/$


S' → • S, $
S → • C C, $
C → • c C, c/d
C → • d, c/d
I0
S
I1
(S' → S • , $
C
S → C • C, $
C → • c C, $
C → • d, $
I2
C
I5
S → C C •, $
c
I36
C → c • C, c/d/$
C → • c C,c/d/$
C → • d,c/d/$
C
c
c
d
I47
d
C → d •, c/d/$
d
I89
C → c C •, c/d/$

LALR Parse Table

	c	d	$	S	C
0	s36	s47		1	2
1			acc		
2	s36	s47			5
36	s36	s47			89
47	r3	r3	r3		
5			r1		
89	r2	r2	r2		

Shift/Reduce Conflict

- We say that we cannot introduce a shift/reduce conflict during the shrink process for the creation of the states of a LALR parser.

Reduce/Reduce Conflict

- But, we may introduce a reduce/reduce conflict during the shrink process for the creation of the states of a LALR parser.

$I_1 : A \rightarrow \alpha\bullet,a$
$B \rightarrow \beta\bullet,b$

$I_2 : A \rightarrow \alpha\bullet,b$
$B \rightarrow \beta\bullet,c$

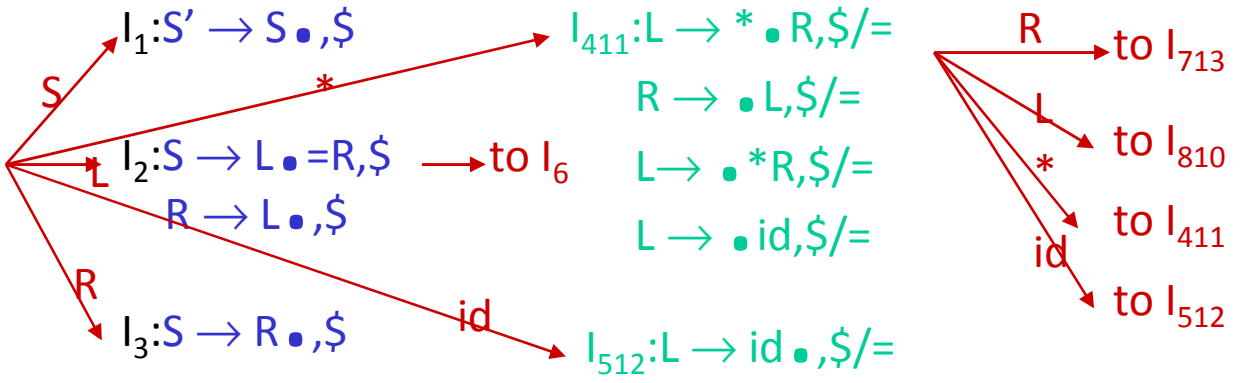
$\Downarrow$

$I_{12} : A \rightarrow \alpha\bullet,a/b$ ➔ reduce/reduce conflict
$B \rightarrow \beta\bullet,b/c$

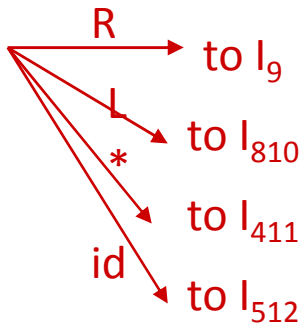
Canonical LALR(1) Collection – Example2

S' → S
1) S → L=R
2) S → R
3) L→ *R
4) L → id
5) R → L

I_0:S' → •S,$
S → •L=R,$
S → •R,$
L → •*R,$/=
L → •id,$/=
R → •L,$



I_6:S → L=•R,$
R → •L,$
L → •*R,$
L → •id,$



I_9:S → L=R•,$

I_{713}:L → *R•,$/=

I_{810}: R → L•,$/=

Same Cores
I_4 and I_{11}
I_5 and I_{12}
I_7 and I_{13}
I_8 and I_{10}

Summary

- LALR parser
  - As simple as SLR and as powerful as CLR
- LR(1) Item construction
  - At some position of the right-hand side with extra terminal symbol as the second component
- LALR parsing table construction
  - Action and goto part
  - Combine similar items with different look ahead characters as one item to reduce number of states

Check your understanding

- Consider the following grammar

  S -> AaAb

  S -> BbBa

  A -> ε

  B -> ε

Construct LALR parsing table for the above grammar.