UNIVERSITY OF ENGINEERING & MANAGEMENT, KOLKATA

Course Name : Compiler Design

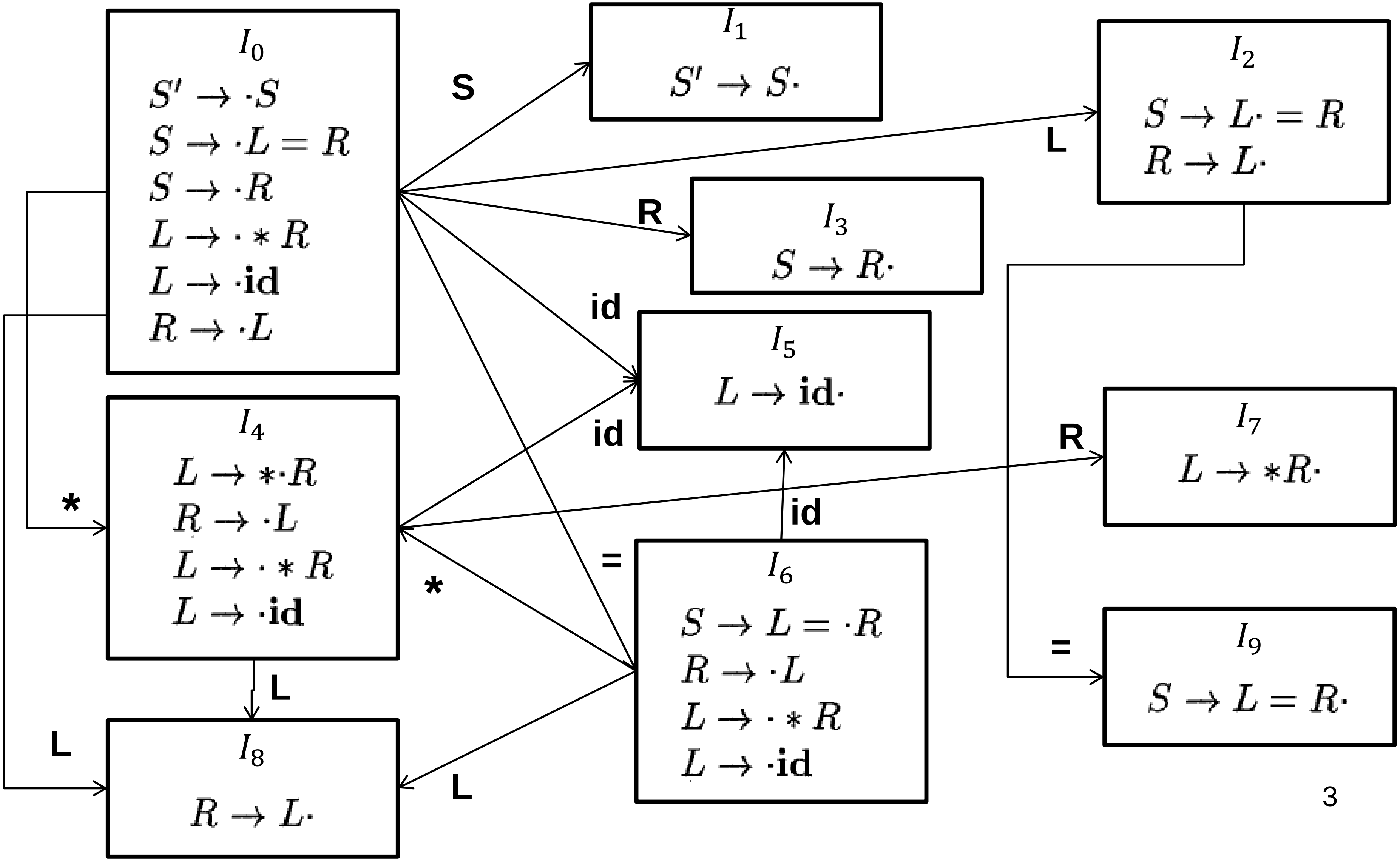
Prof. Sankhadeep Chatterjee

SLR(1) example

Consider the following grammar and construct the SLR parsing table:

$$
\begin{aligned}
S &\rightarrow L = R \mid R \\
L &\rightarrow *R \mid \mathbf{id} \\
R &\rightarrow L
\end{aligned}
$$

LR(0) Automaton



SLR Parsing Table

	ACTION				GOTO		
STATE	=	*	***id***	$	***S***	***L***	***R***
0		s4	s5		1	2	3
1				acc			
2	s6,r6			r6			
3				r3			
4		s4	s5			8	7
5	r5			r5			
6		s4	s5			8	9
7	r4			r4			
8	r6			r6			
9				r2			

Example parsing

- There is both a shift and a reduce entry in action[2,=].
  - Therefore state 2 has a shift- reduce conflict on symbol “=“
  - However, the grammar is not ambiguous.
- Parse $\boldsymbol{id} = \boldsymbol{id}$ assuming reduce action is taken in [2,=]

Stack	input	action
0	id=id $	shift 5
0 5	=id $	reduce by L -> id
0 2	=id $	reduce by R -> L
0 3	=id $	error

Example parsing

- if shift action is taken in [2,=]

Stack	input	action
0	id=id$	shift 5
0 5	=id$	reduce by L -> id
0 2	=id$	shift 6
0 2 6	id$	shift 5
0 2 6 5	$	reduce by L->id
0 2 6 8	$	reduce by R -> L
0 2 6 9	$	reduce by S -> L=R
0 1	$	ACCEPT

Limitation of SLR

- No sentential form of this grammar can start with R=…
- However, the reduce action in action[2,=] generates a sentential form starting with R=
- Therefore, the reduce action is incorrect
- In SLR parsing method state i calls for reduction on symbol “a”, by rule $A \rightarrow \alpha$ if I_i contains $[A \rightarrow \alpha.]$ and “a” is in $FOLLOW(A)$
- However, when state I appears on the top of the stack, the viable prefix $\beta\alpha$ on the stack may be such that βA can not be followed by symbol “a” in any right sentential form
- Thus, the reduction by the rule $A \rightarrow \alpha$ on symbol “a” is invalid
- SLR parsers cannot remember the left context

Canonical LR Parsing

- Carry extra information in the state so that wrong reductions by $A \rightarrow \alpha$ will be ruled out
- Redefine LR items to include a terminal symbol as a second component (look ahead symbol)
- The general form of the item becomes $[A \rightarrow \alpha.\beta, a]$ which is called LR(1) item.
- Item $[A \rightarrow \alpha., a]$ calls for reduction only if next input is a. The set of symbols "a"s will be a subset of $FOLLOW(A)$.

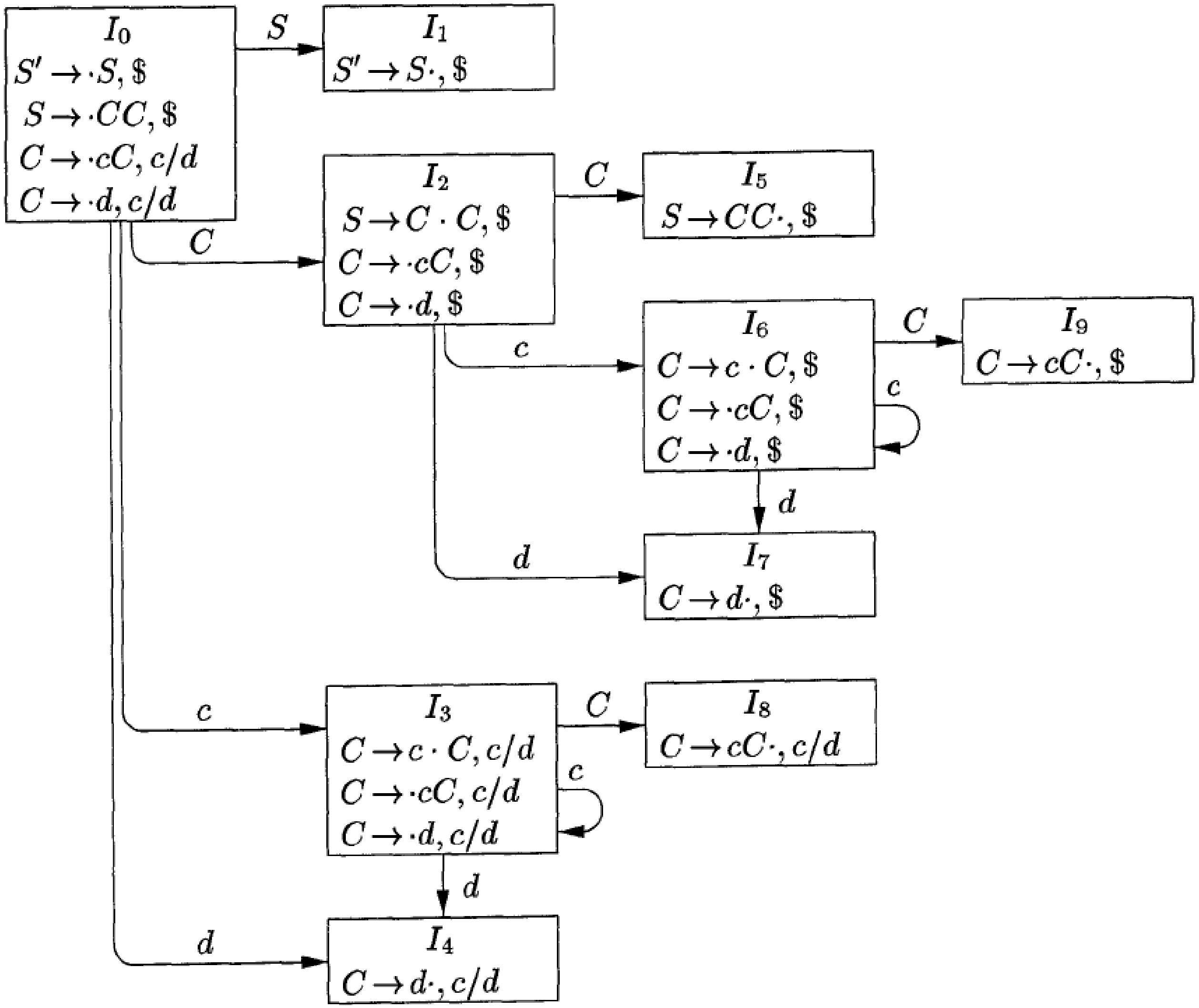
Closure of LR(1) Items

```
repeat
    for ( each item [A → α·Bβ, a] in I )
        for ( each production B → γ in G' )
            for ( each terminal b in FIRST(βa) )
                add [B → ·γ, b] to set I;
until no more items are added to I;
return I;
```

Definitions of GOTO and construction of LR(1) sets items is same as SLR.

Construct the LR(1) automaton for the following grammar:

$$S' \to S$$
$$S \to CC$$
$$C \to cC \mid d$$



I_0
$S' \to \cdot S, \$$
$S \to \cdot CC, \$$
$C \to \cdot cC, c/d$
$C \to \cdot d, c/d$
S
I_1
$S' \to S\cdot, \$$
C
I_2
$S \to C \cdot C, \$$
$C \to \cdot cC, \$$
$C \to \cdot d, \$$
C
I_5
$S \to CC\cdot, \$$
c
I_6
$C \to c \cdot C, \$$
$C \to \cdot cC, \$$
$C \to \cdot d, \$$
C
I_9
$C \to cC\cdot, \$$
c
d
d
I_7
$C \to d\cdot, \$$
c
I_3
$C \to c \cdot C, c/d$
$C \to \cdot cC, c/d$
$C \to \cdot d, c/d$
C
I_8
$C \to cC\cdot, c/d$
c
d
d
I_4
$C \to d\cdot, c/d$

Canonical LR(1) Parsing Table Generation

1. Construct $C' = \{I_0, I_1, \cdots, I_n\}$, the collection of sets of LR(1) items for G'.

2. State i of the parser is constructed from I_i. The parsing action for state i is determined as follows.

   (a) If $[A \to \alpha{\cdot}a\beta, b]$ is in I_i and $\text{GOTO}(I_i, a) = I_j$, then set $\text{ACTION}[i, a]$ to "shift j." Here a must be a terminal.

   (b) If $[A \to \alpha\cdot, a]$ is in I_i, $A \neq S'$, then set $\text{ACTION}[i, a]$ to "reduce $A \to \alpha$."

   (c) If $[S' \to S\cdot, \$]$ is in I_i, then set $\text{ACTION}[i, \$]$ to "accept."

   If any conflicting actions result from the above rules, we say the grammar is not LR(1). The algorithm fails to produce a parser in this case.

3. The goto transitions for state i are constructed for all nonterminals A using the rule: If $\text{GOTO}(I_i, A) = I_j$, then $\text{GOTO}[i, A] = j$.

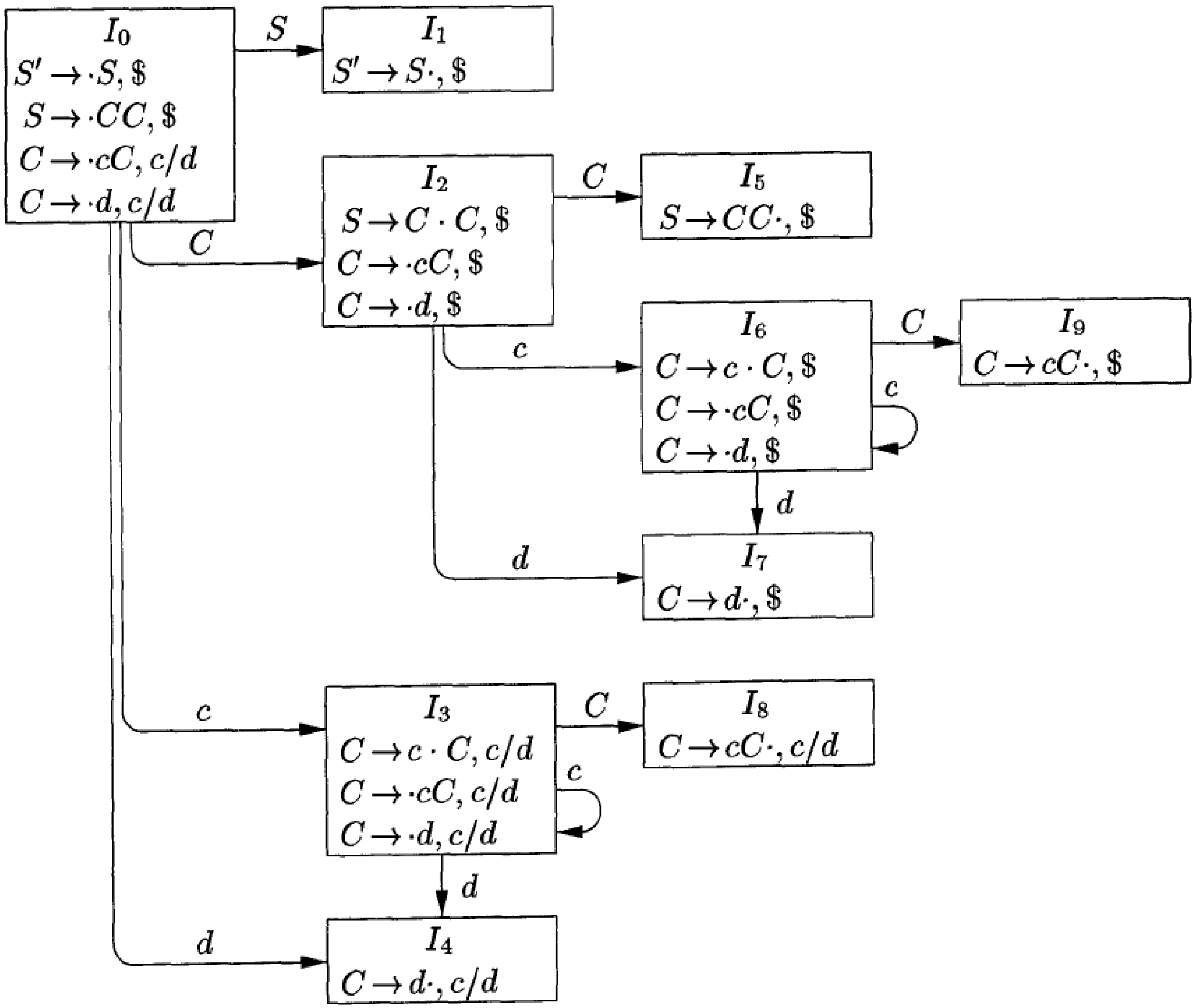
4. All entries not defined by rules (2) and (3) are made "error."

5. The initial state of the parser is the one constructed from the set of items containing $[S' \to \cdot S, \$]$.

Canonical LR(1) Parsing Table Example

Construct the Canonical LR(1) parsing table for the following grammar:

$$S' \rightarrow S$$
$$S \rightarrow CC$$
$$C \rightarrow cC \mid d$$



Canonical LR(1) Parsing Table Example

	ACTION			GOTO	
State	***c***	***d***	$	***S***	***C***
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

- 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

Look Ahead LR parsers

- Consider a pair of similar looking states (same kernel and different look aheads) in the set of LR(1) items

$I_4 : \{C \rightarrow d., c/d\}$ $\quad$ $I_7 : \{C \rightarrow d., \$\}$

- Replace I_4 and I_7 by a new state I_{47} consisting of $(C \rightarrow d., c/d/\$)$

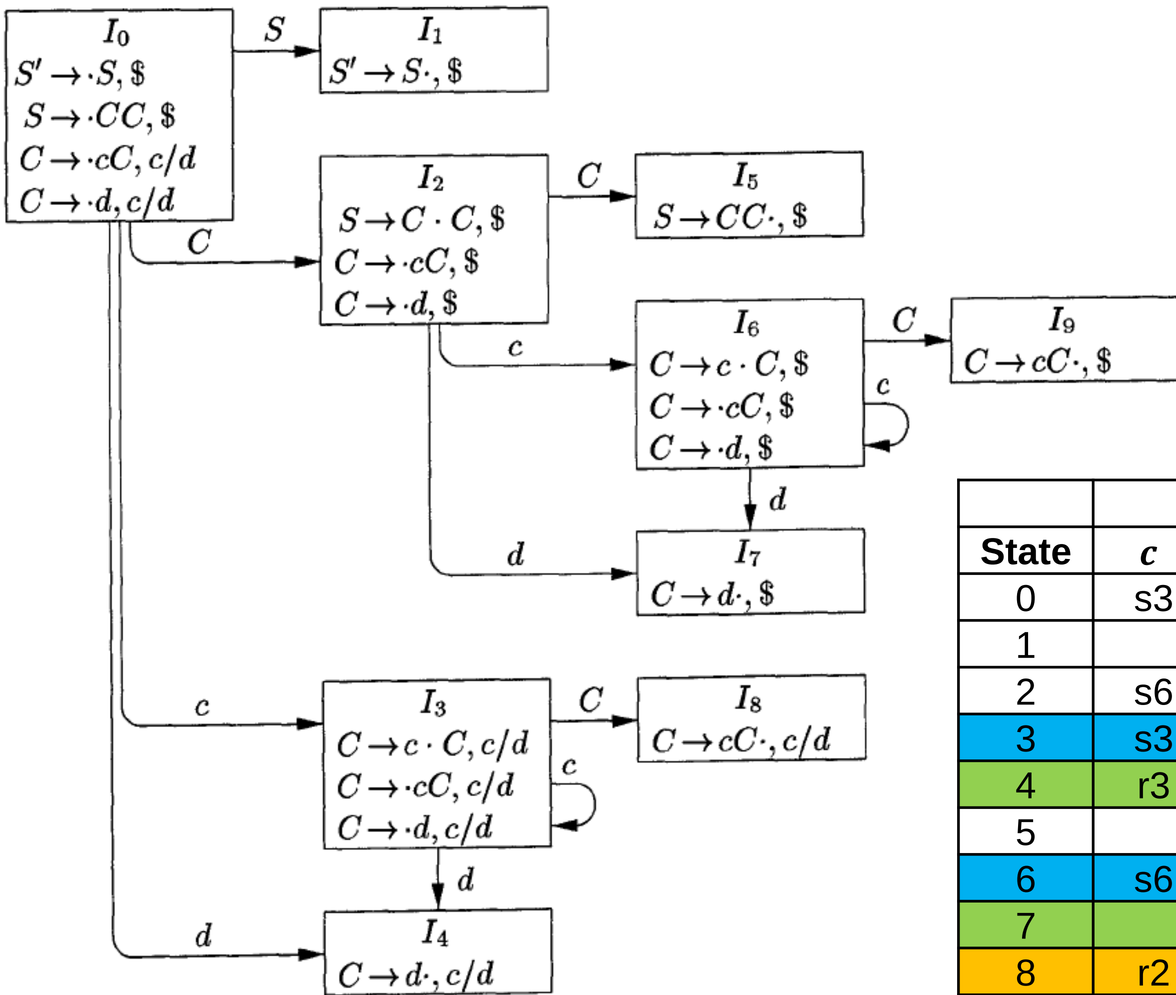
- Similarly I_3 & I_6 and I_8 & I_9 form pairs

- Merge LR(1) items having the same core

LALR(1) parsing table generation

- Construct C = $\{I_0,\ldots\ldots,I_n\}$ set of LR(1) items
- For each core present in LR(1) items find all sets having the same core and replace these sets by their union
- Let C' = $\{J_0,\ldots\ldots.,J_m\}$ be the resulting set of items Construct action table as was done earlier
- Let $J = I_1 \cup I_2 \ldots\ldots\cup I_k$ since I_1 , $I_2\ldots\ldots.$, 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) then GOTO (J,X) = K

LALR(1) parsing table generation example



Construct the LALR parsing table from the following Canonical LR(1) collection and parsing table

	ACTION			GOTO	
State	***c***	***d***	$	***S***	***C***
0	s3	s4		1	2
1			acc		
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) parsing table generation example

	ACTION			GOTO	
State	***c***	***d***	$	***S***	***C***
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

State	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		