

# Sub: Compiler Construction

## Syntax Analysis PART 5

Compiled for: 7th Sem, CE, DDU

Compiled by: Niyati J. Buch

# Topics Covered

- Bottom-up parsing
  - CLR
    - [Constructing LR\(1\) item set](#)
    - [Constructing LR\(1\) parsing table](#)
  - LALR
    - [LALR\(1\) item sets](#)
    - [LALR\(1\) parsing table](#)
    - [Example to demonstrate reduce/reduce conflict after merger](#)
    - [Show that the grammar is LALR\(1\) but not SLR\(1\)](#)
    - [Show that the grammar is LR\(1\) but not LALR\(1\)](#)
  - Ambiguous grammar
    - [Expression grammar](#)
    - [Dangling else grammar](#)
  - [Error Recovery](#)

# More Powerful LR Parsers

1. The "**canonical-LR**" or just "LR" method, which makes full use of the lookahead symbol(s).
  - This method uses a large set of items, called the **LR(1) items**.
2. The "**lookahead-LR**" or "LALR" method, which is based on the **LR(0) sets of items**, and has many fewer states than typical parsers based on the LR(1) items.
  - By carefully introducing lookaheads into the LR(0) items, we can handle many more grammars with the LALR method than with the SLR method, and build parsing tables that are no bigger than the SLR tables.
  - **LALR is the method of choice in most situations.**

# LR(1) Item

- The extra information is incorporated into the state by redefining items to include a terminal symbol as a second component. The general form of an item becomes  $[A \rightarrow \alpha \cdot \beta, a]$ , where  $A \rightarrow \alpha \beta$  is a production and **a is a terminal or the right endmarker \$**. We call such an object **an LR(1) item**. The 1 refers to the length of the second component, called the lookahead of the item.
- The lookahead has no effect in an item of the form  $[A \rightarrow \alpha \cdot \beta, a]$  where  $\beta$  is not  $\epsilon$ , but an item of the form  $[A \rightarrow \alpha \cdot, a]$  calls for a reduction by  $A \rightarrow \alpha$  only if the next input symbol is a. Thus, we are compelled to reduce by  $A \rightarrow \alpha$  only on those input symbols a for which  $[A \rightarrow \alpha \cdot, a]$  is an LR(1) item in the state on top of the stack.

# Constructing LR(1) Sets of Items

```
SetOfItems CLOSURE( $I$ ) {  
    repeat  
        for ( each item  $[A \rightarrow \alpha \cdot B \beta, a]$  in  $I$  )  
            for ( each production  $B \rightarrow \gamma$  in  $G'$  )  
                for ( each terminal  $b$  in FIRST( $\beta a$ ) )  
                    add  $[B \rightarrow \cdot \gamma, b]$  to set  $I$ ;  
    until no more items are added to  $I$ ;  
    return  $I$ ;  
}
```

# Constructing LR(1) Sets of Items

```
SetOfItems GOTO( $I, X$ ) {  
    initialize  $J$  to be the empty set;  
    for ( each item  $[A \rightarrow \alpha \cdot X \beta, a]$  in  $I$  )  
        add item  $[A \rightarrow \alpha X \cdot \beta, a]$  to set  $J$ ;  
    return CLOSURE( $J$ );  
}
```

# Constructing LR(1) Sets of Items

```
void items( $G'$ ) {  
    initialize  $C$  to  $\{\text{CLOSURE}(\{[S' \rightarrow \cdot S, \$]\})\}$ ;  
    repeat  
        for ( each set of items  $I$  in  $C$  )  
            for ( each grammar symbol  $X$  )  
                if (  $\text{GOTO}(I, X)$  is not empty and not in  $C$  )  
                    add  $\text{GOTO}(I, X)$  to  $C$ ;  
    until no new sets of items are added to  $C$ ;  
}
```

# Constructing LR(1) Sets of Items

- Consider the following augmented grammar.

$S' \rightarrow S$

$S \rightarrow C C$

$C \rightarrow c C \mid d$

- The general form of an item becomes  $[A \rightarrow \alpha \cdot \beta, a]$ ,  
where  $A \rightarrow \alpha \beta$  is a production and **a is a terminal or the right endmarker \$**
- For all productions  $B \rightarrow \gamma$  add  $[B \rightarrow \cdot \gamma, b]$  where  $b$  is  $\text{FIRST}(\beta a)$



# Constructing LR(1) Sets of Items

- Consider the following augmented grammar.

$S' \rightarrow S$

$S \rightarrow CC$

$C \rightarrow cC \mid d$

- The general form of an item becomes  $[A \rightarrow \alpha \cdot \beta, a]$ , where  $A \rightarrow \alpha \beta$  is a production and **a is a terminal or the right endmarker \$**
- $[S' \rightarrow \cdot S, \$]$  compare with  $[A \rightarrow \alpha \cdot B \beta, a]$ , A is  $S'$ ,  $\alpha$  is  $\epsilon$ , B is S,  $\beta$  is  $\epsilon$  and a is \$
- For all productions  $B \rightarrow \gamma$  i.e.  $S \rightarrow CC$ , add  $[B \rightarrow \cdot \gamma, b]$  where  $b$  is  $\text{FIRST}(\beta a) = \text{FIRST}(\epsilon \$) = \{\$ \}$ . Hence,  $[S \rightarrow \cdot CC, \$]$  is added.
- $[S \rightarrow \cdot CC, \$]$  compare with  $[A \rightarrow \alpha \cdot B \beta, a]$ , A is S,  $\alpha$  is  $\epsilon$ , B is C,  $\beta$  is C and a is \$
- For all productions  $B \rightarrow \gamma$  i.e.  $C \rightarrow cC \mid d$ , add  $[B \rightarrow \cdot \gamma, b]$  where  $b$  is  $\text{FIRST}(\beta a) = \text{FIRST}(C \$) = \{c, d\}$ . Hence,  $[C \rightarrow \cdot cC, c/d]$  and  $[C \rightarrow \cdot d, c/d]$  are added.

# Constructing LR(1) Sets of Items

- Consider the following augmented grammar.

$S' \rightarrow S$

$S \rightarrow CC$

$C \rightarrow cC \mid d$

|                               |       |
|-------------------------------|-------|
|                               | $I_0$ |
| $S' \rightarrow \cdot S, \$$  |       |
| $S \rightarrow \cdot CC, \$$  |       |
| $C \rightarrow \cdot cC, c/d$ |       |
| $C \rightarrow \cdot d, c/d$  |       |

- The general form of an item becomes  $[A \rightarrow \alpha \cdot \beta, a]$ , where  $A \rightarrow \alpha \beta$  is a production and  $a$  is a terminal or the right endmarker  $\$$
- $[S' \rightarrow \cdot S, \$]$  compare with  $[A \rightarrow \alpha \cdot B \beta, a]$ ,  $A$  is  $S'$ ,  $\alpha$  is  $\epsilon$ ,  $B$  is  $S$ ,  $\beta$  is  $\epsilon$  and  $a$  is  $\$$
- For all productions  $B \rightarrow \gamma$  i.e.  $S \rightarrow CC$ , add  $[B \rightarrow \cdot \gamma, b]$  where  $b$  is  $\text{FIRST}(\beta a) = \text{FIRST}(\epsilon \$) = \{\$ \}$ . Hence,  $[S \rightarrow \cdot CC, \$]$  is added.
- $[S \rightarrow \cdot CC, \$]$  compare with  $[A \rightarrow \alpha \cdot B \beta, a]$ ,  $A$  is  $S$ ,  $\alpha$  is  $\epsilon$ ,  $B$  is  $C$ ,  $\beta$  is  $C$  and  $a$  is  $\$$
- For all productions  $B \rightarrow \gamma$  i.e.  $C \rightarrow cC \mid d$ , add  $[B \rightarrow \cdot \gamma, b]$  where  $b$  is  $\text{FIRST}(\beta a) = \text{FIRST}(C\$) = \{c, d\}$ . Hence,  $[C \rightarrow \cdot cC, c/d]$  and  $[C \rightarrow \cdot d, c/d]$  are added.

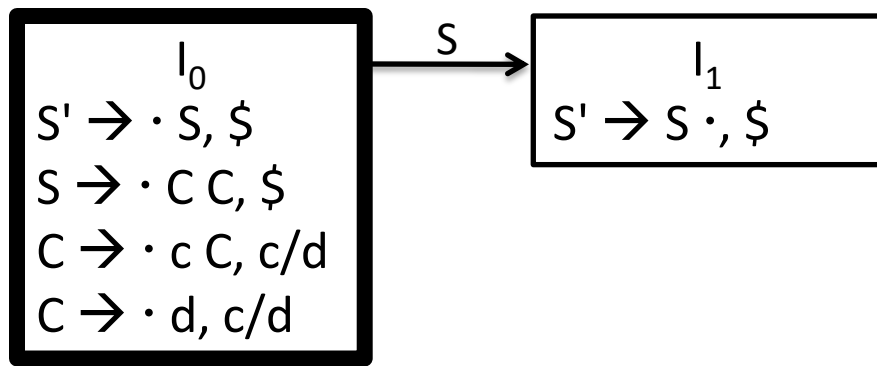
$I_0$

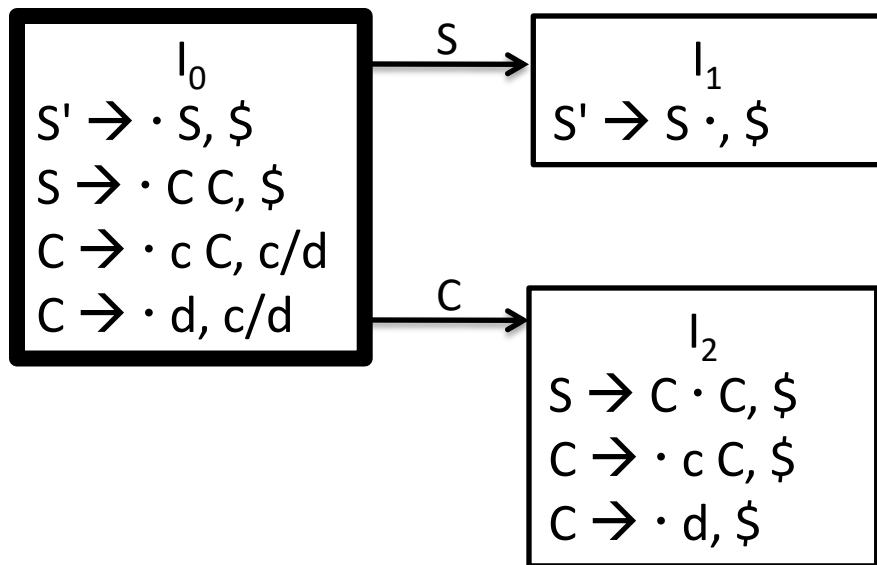
$S' \rightarrow \cdot S, \$$

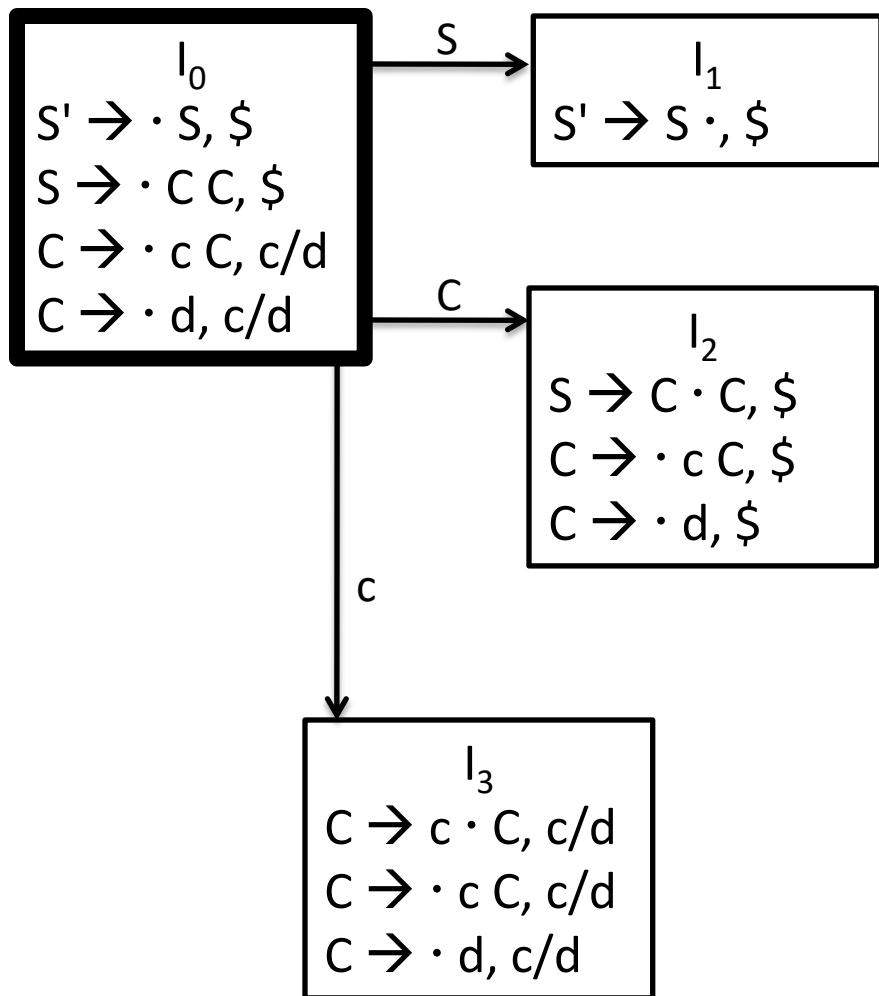
$S \rightarrow \cdot C C, \$$

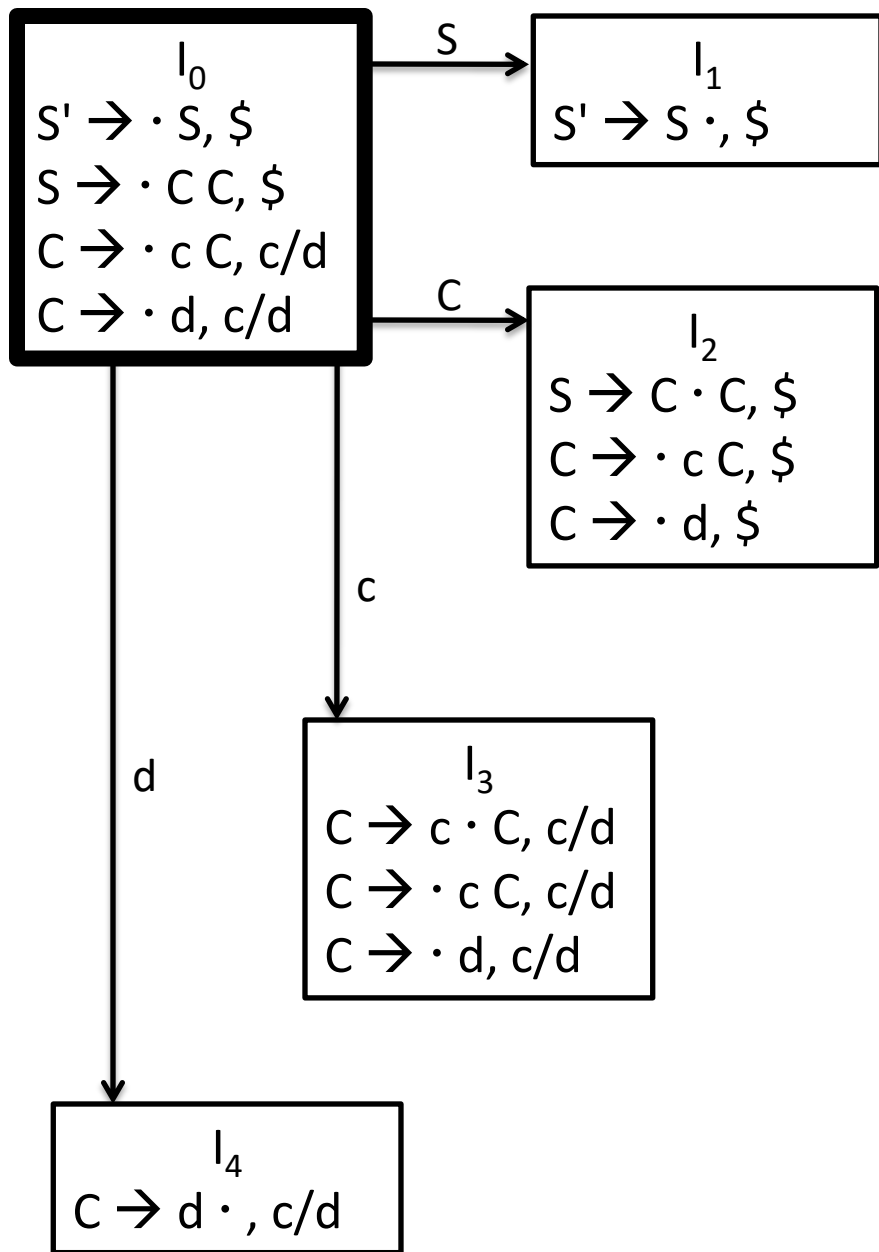
$C \rightarrow \cdot c C, c/d$

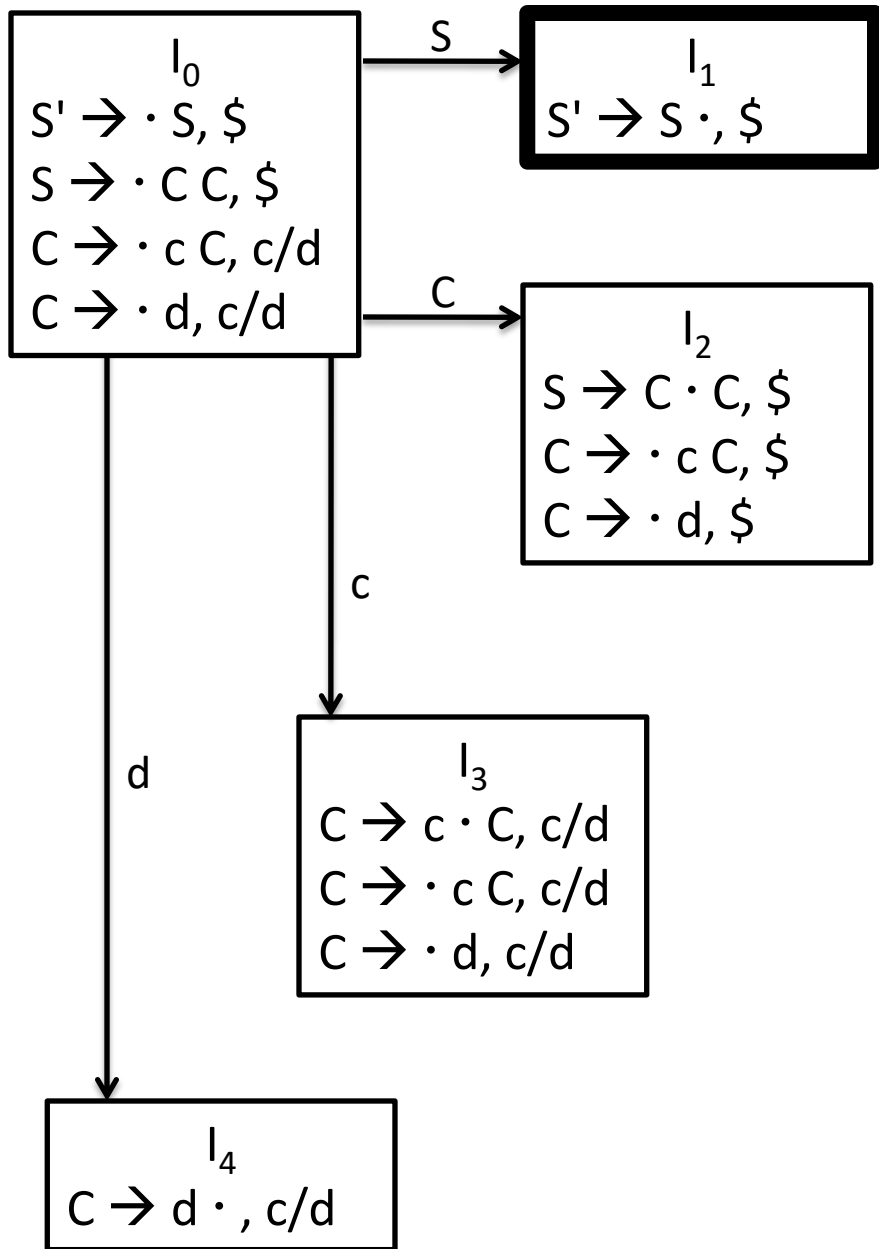
$C \rightarrow \cdot d, c/d$



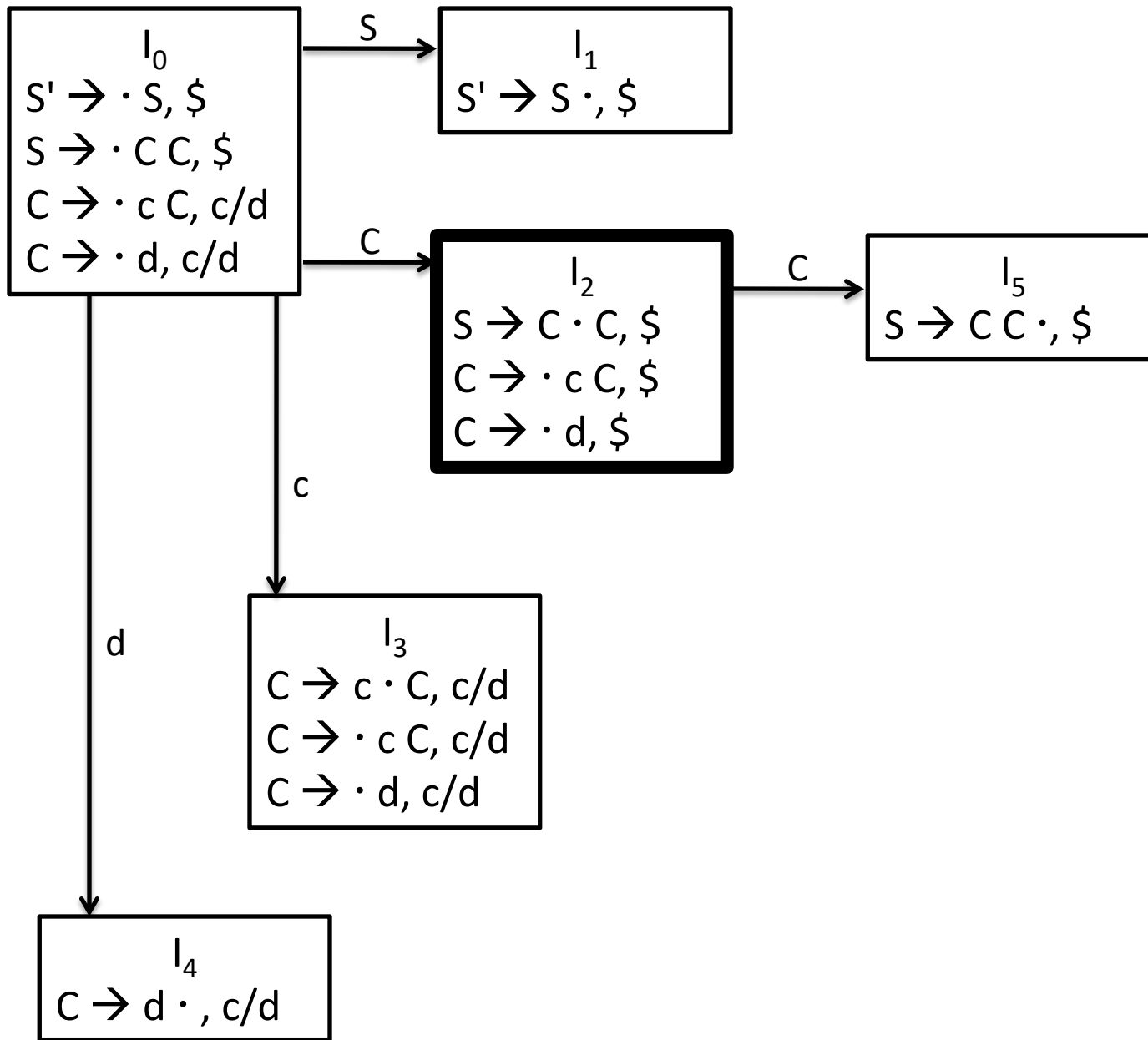


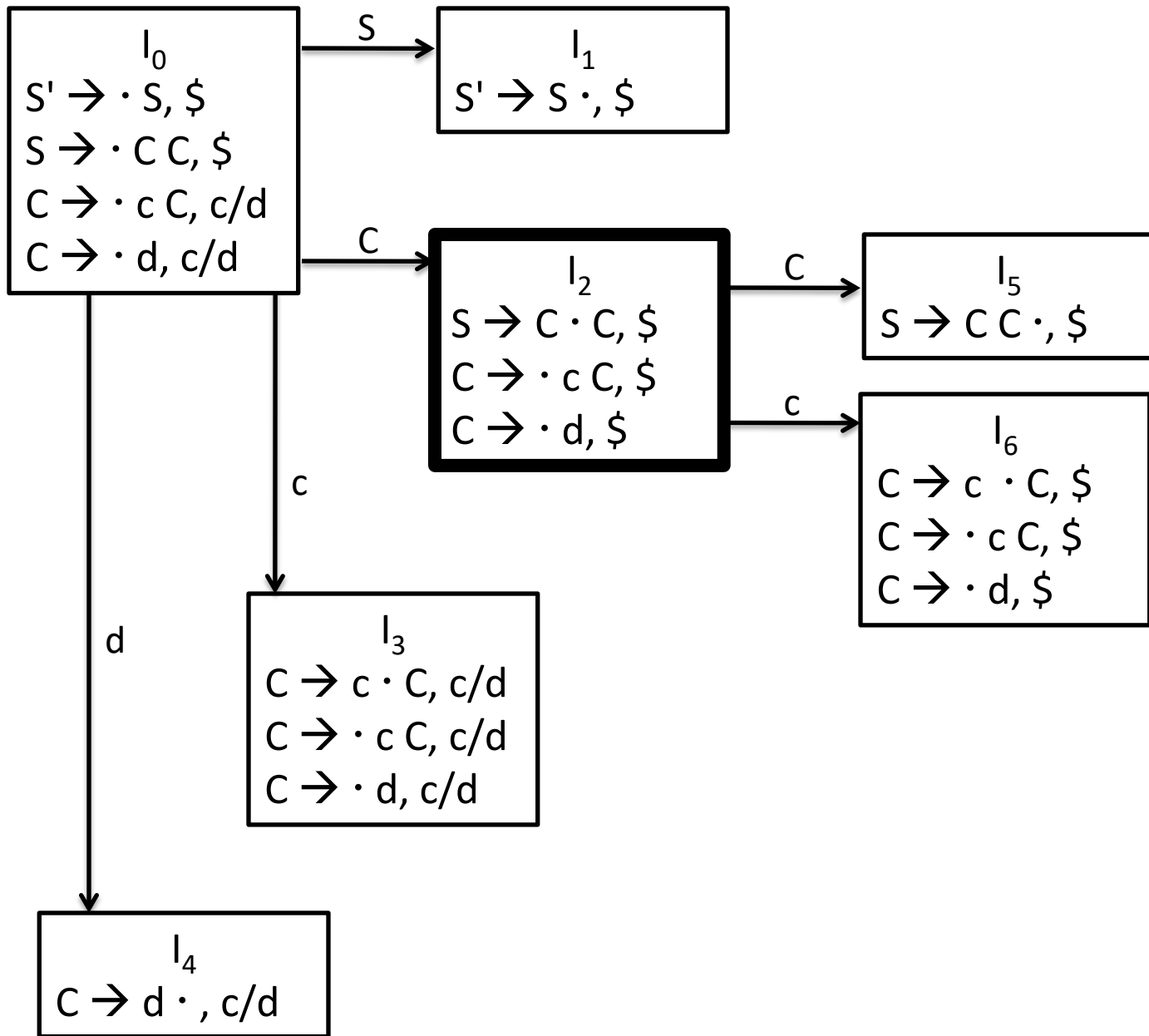


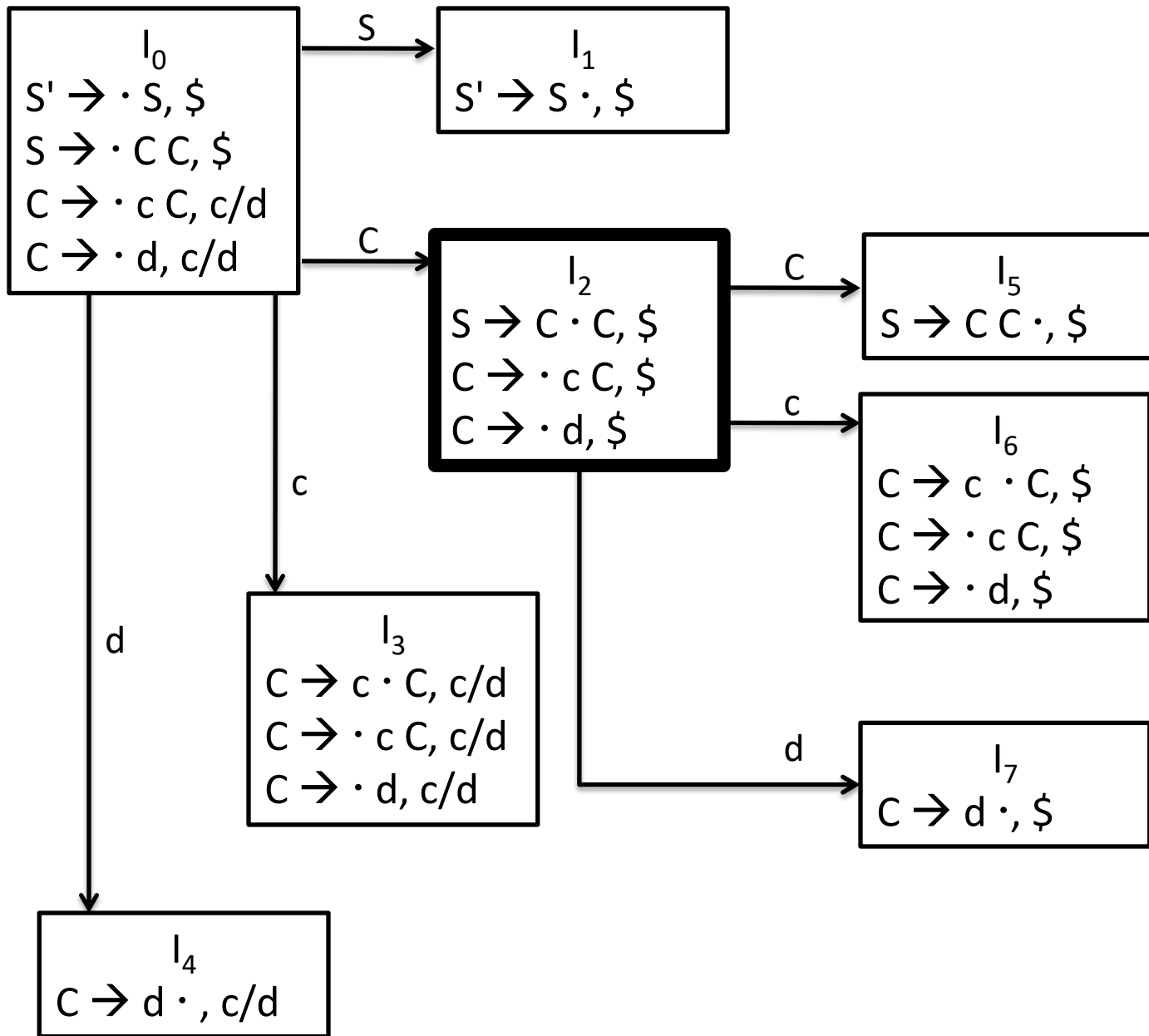


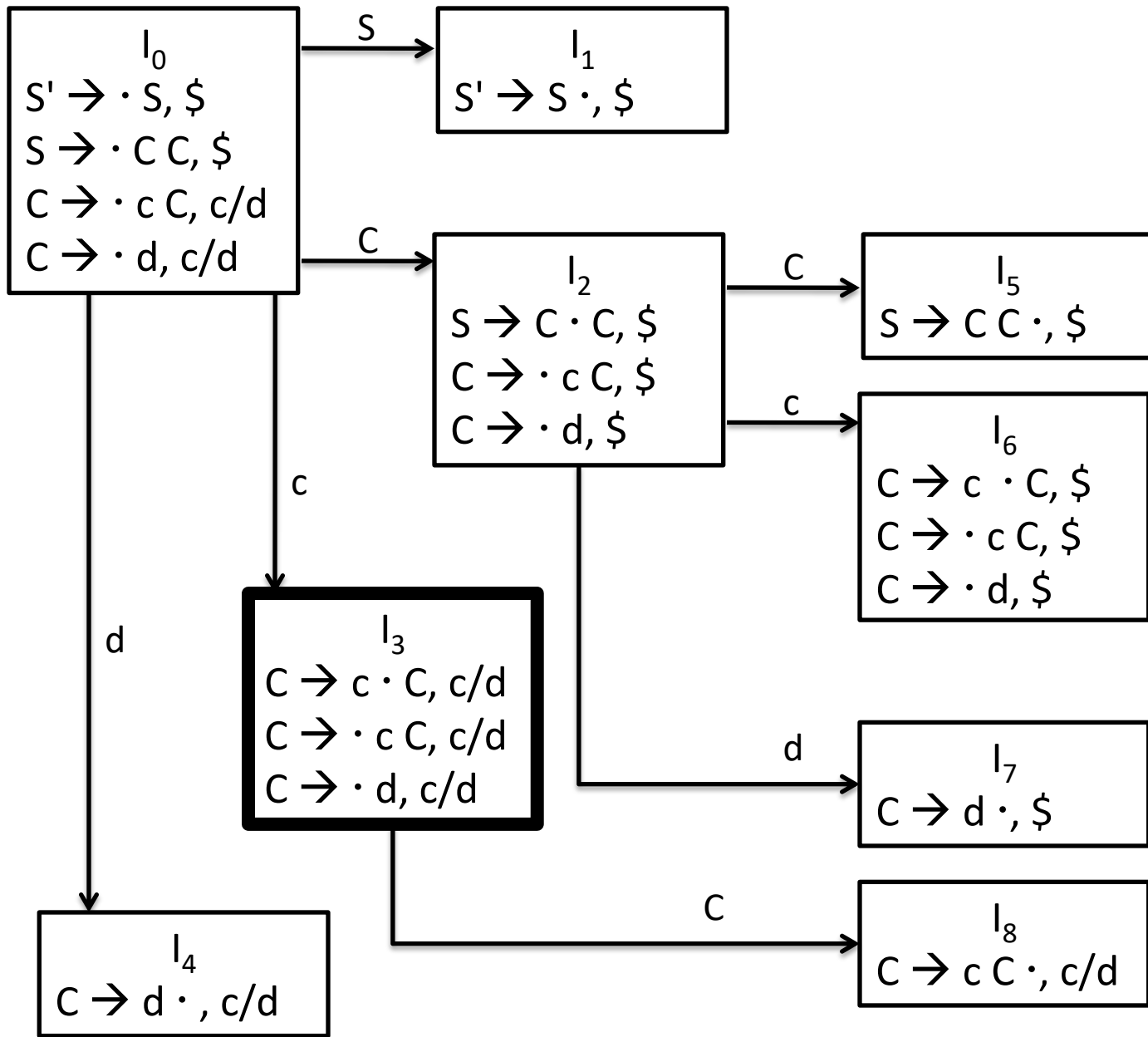


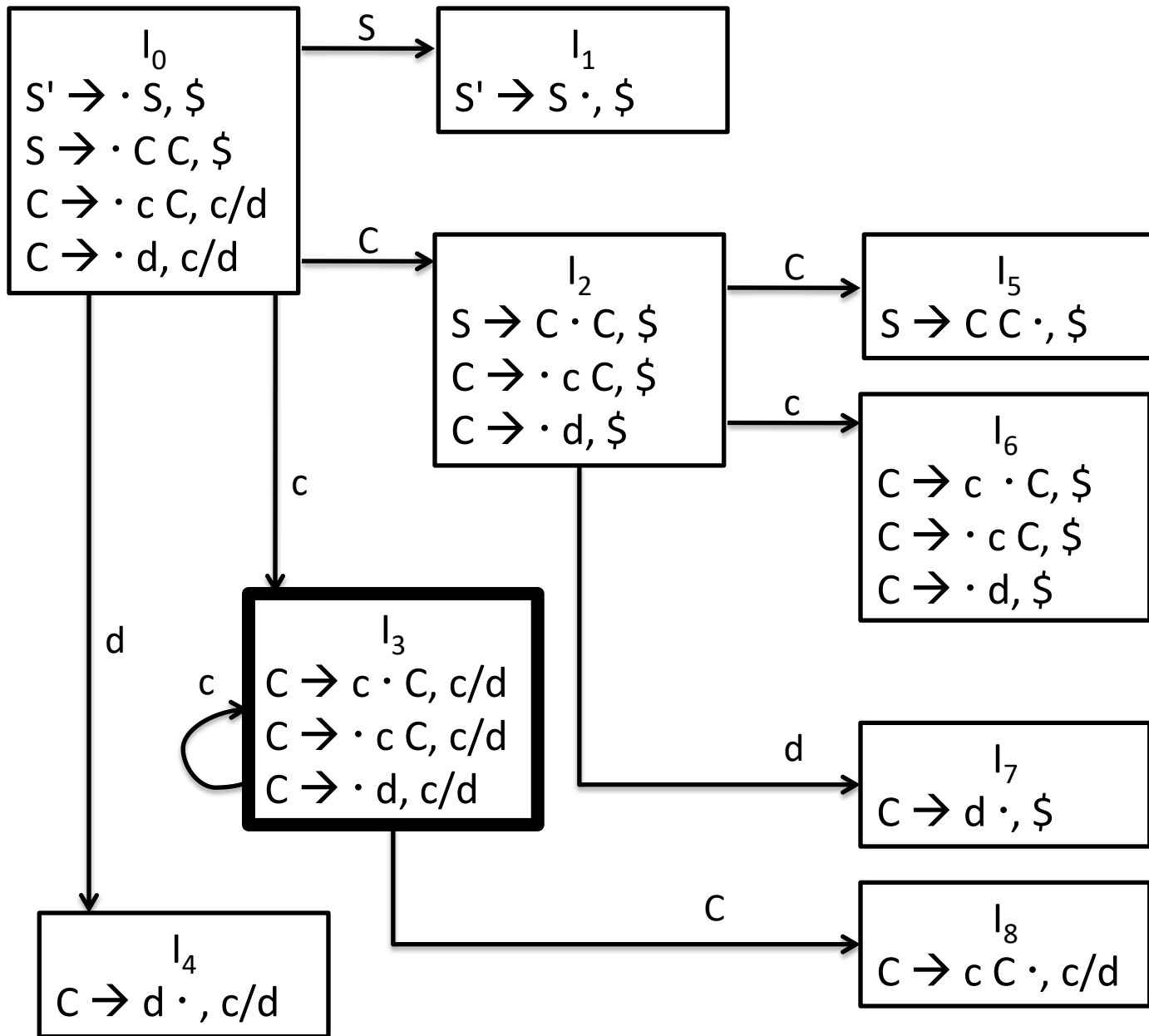


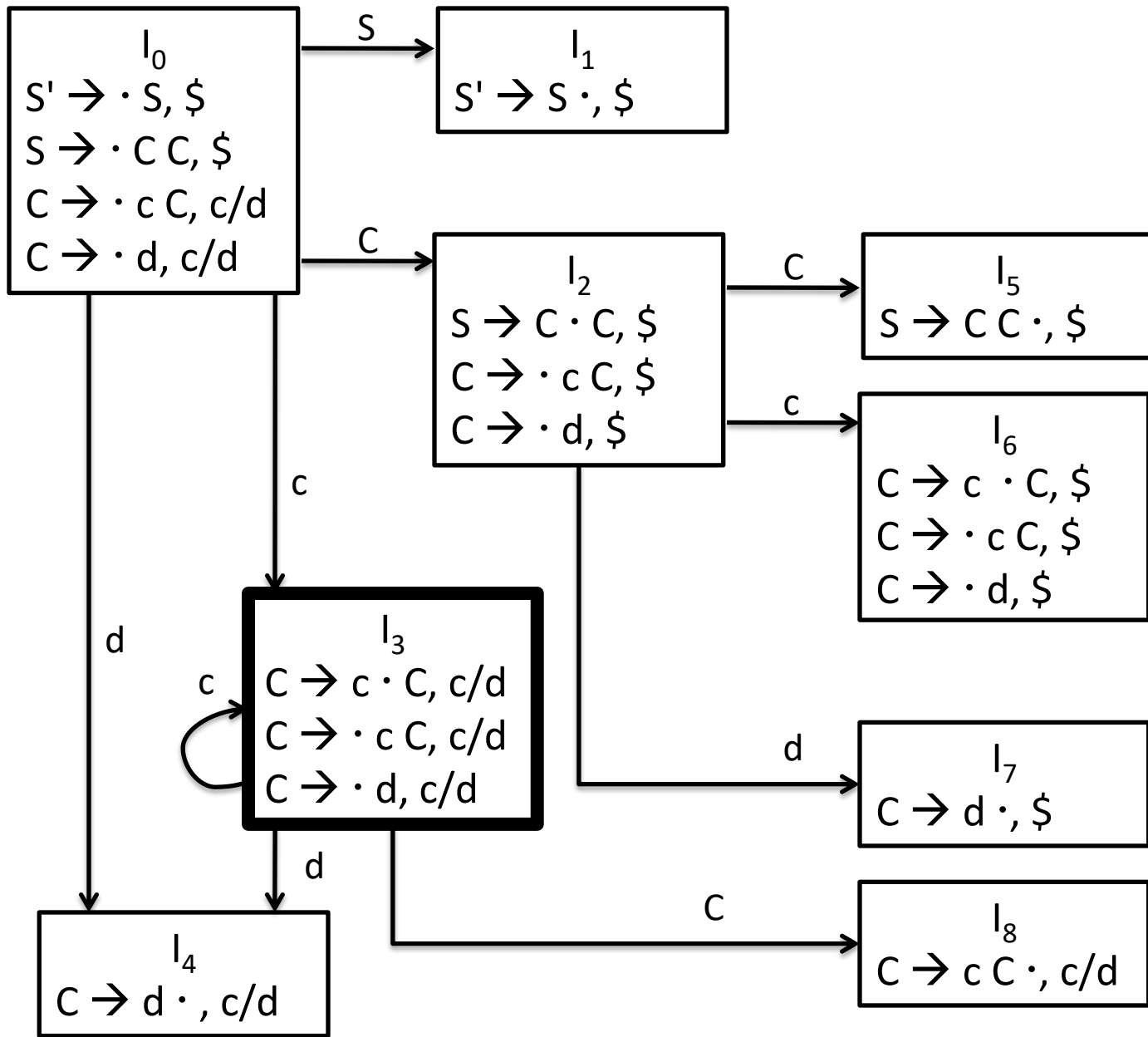


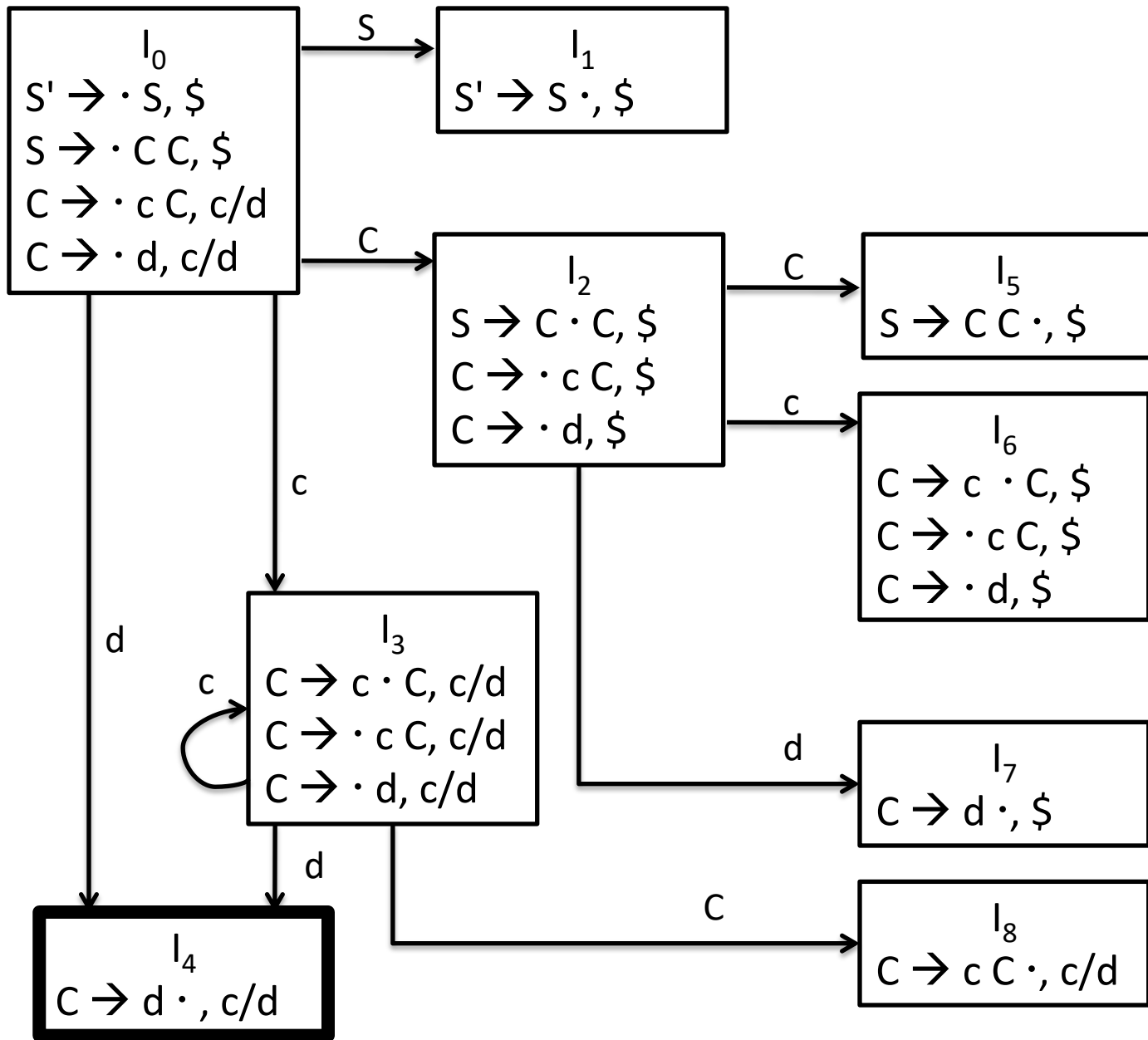


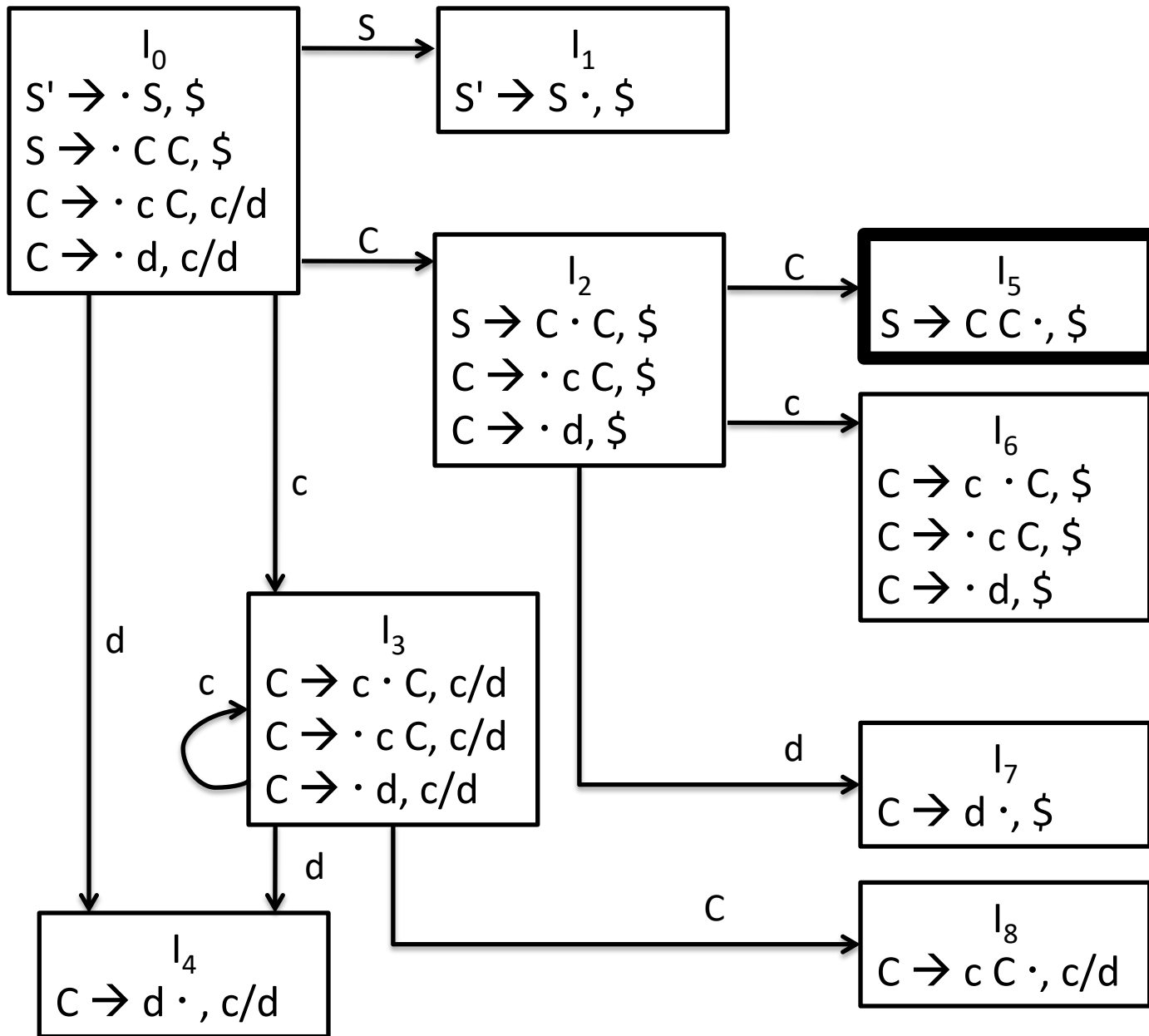




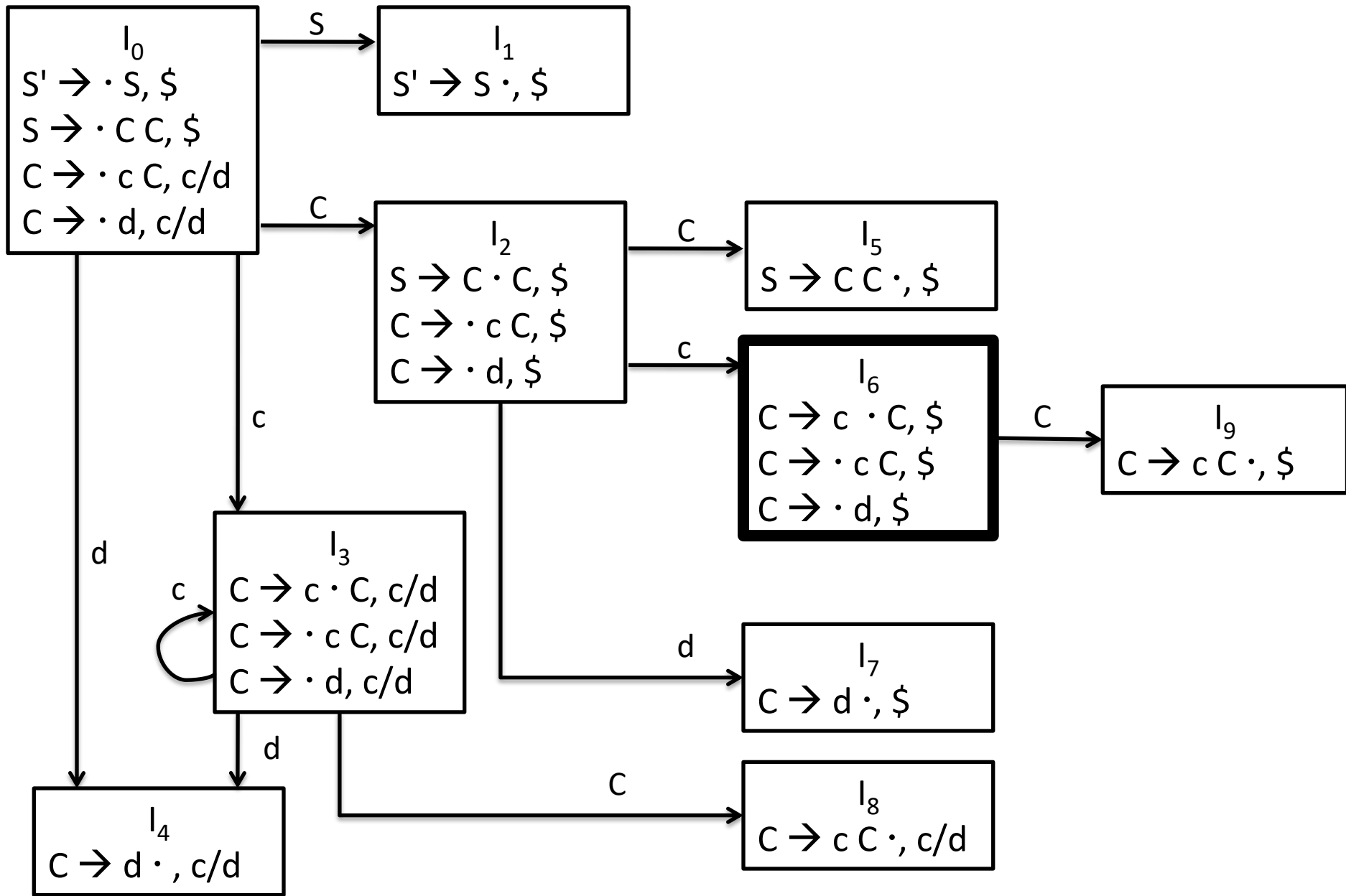


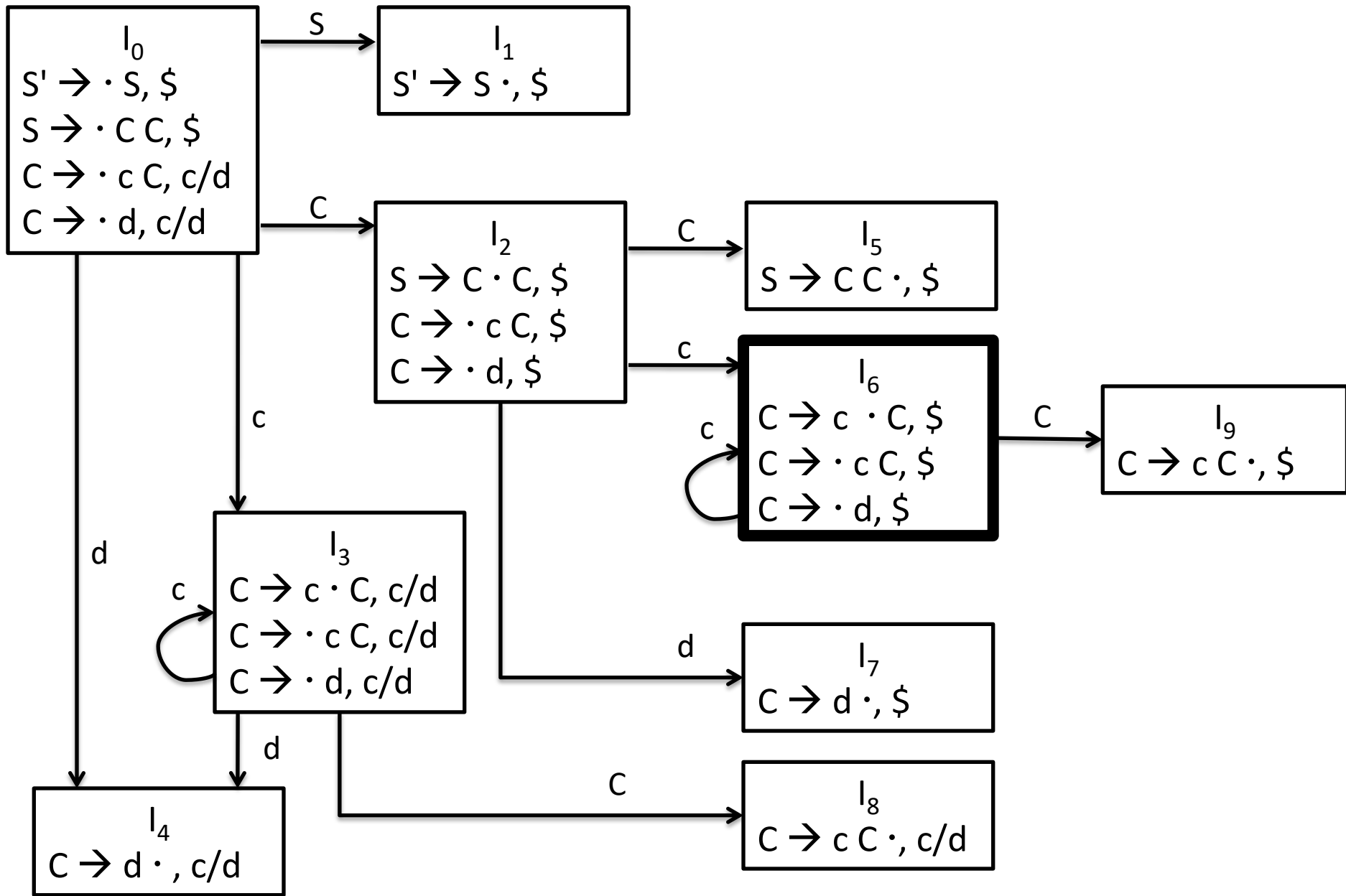


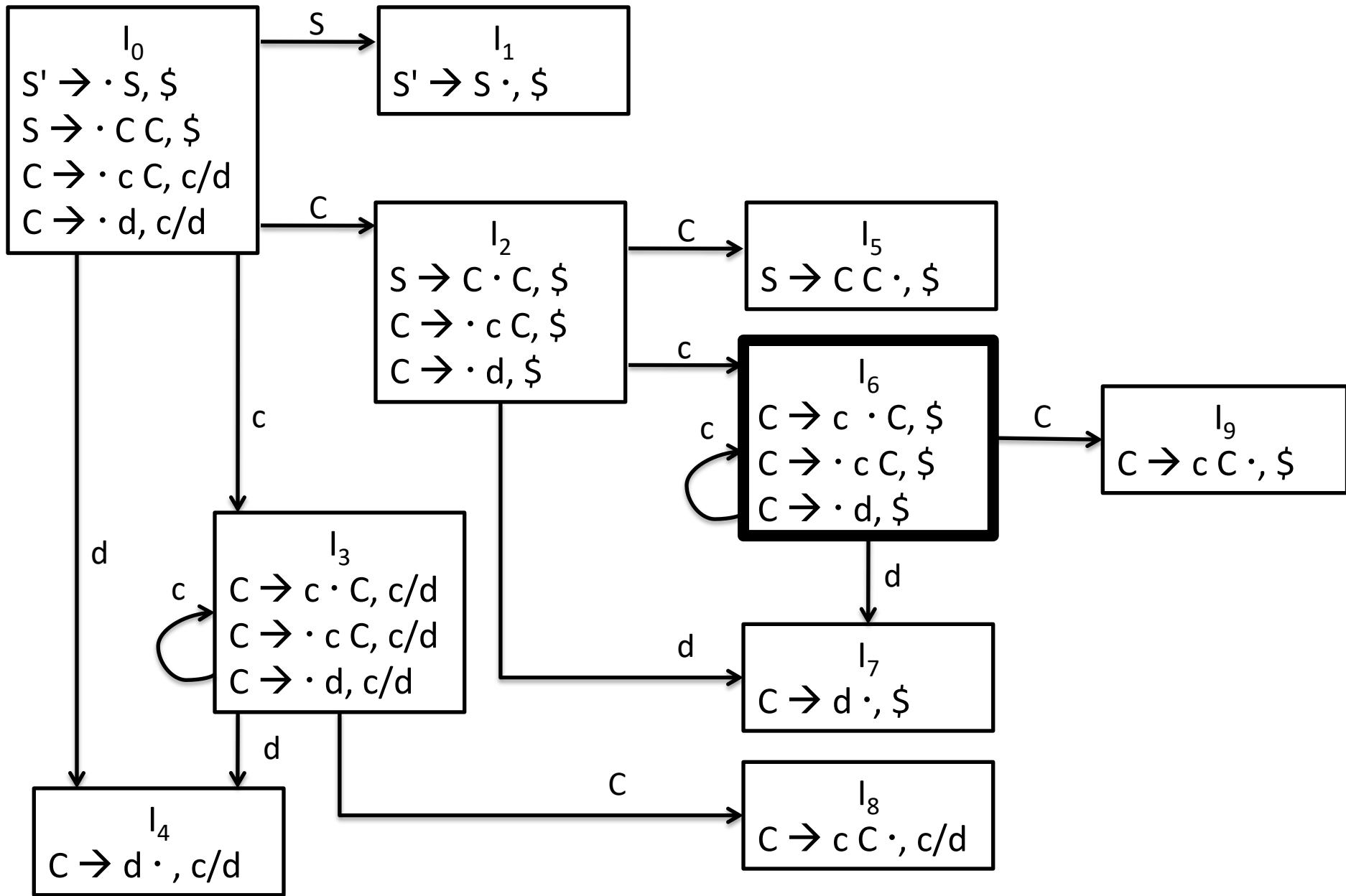


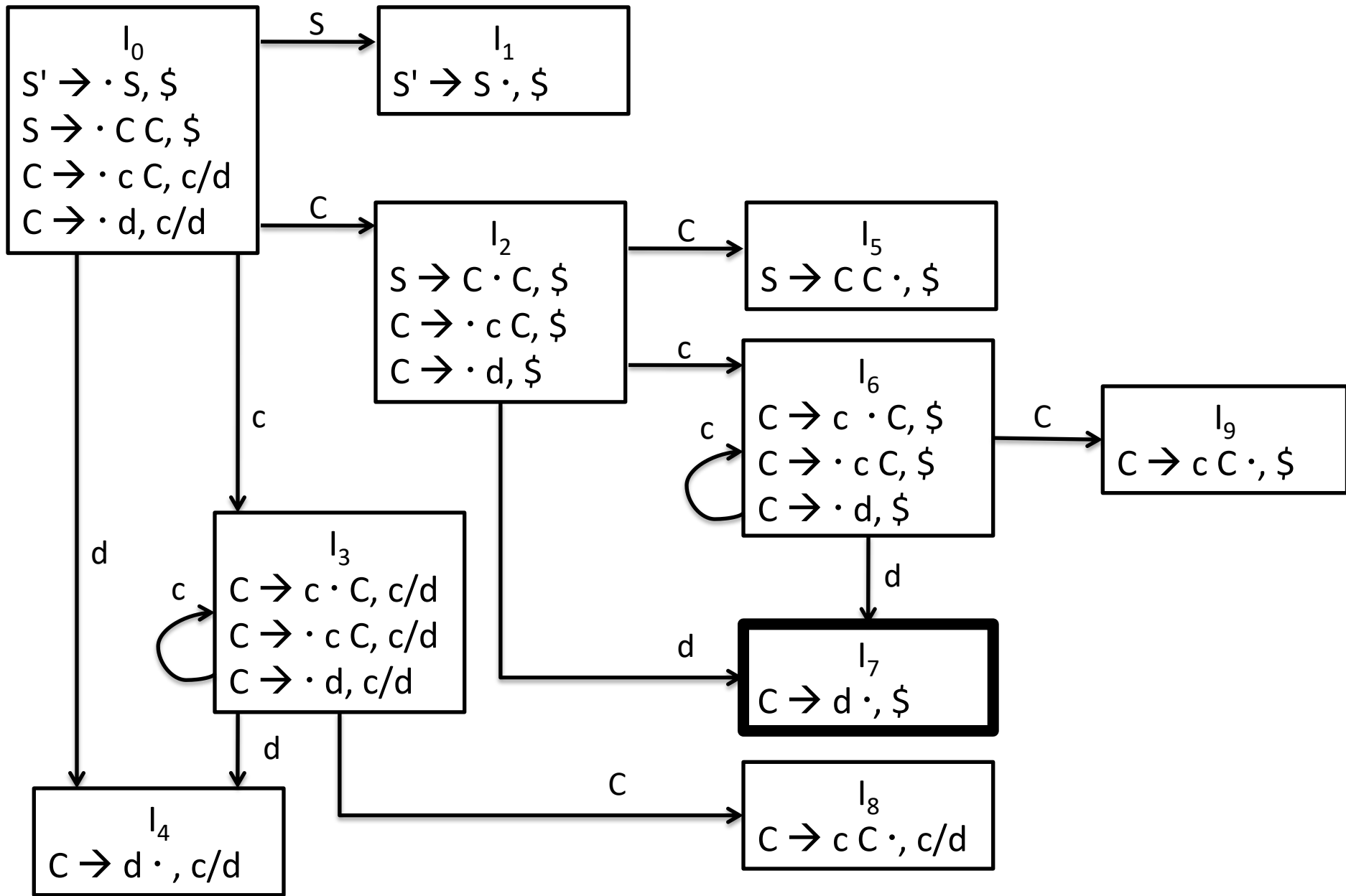


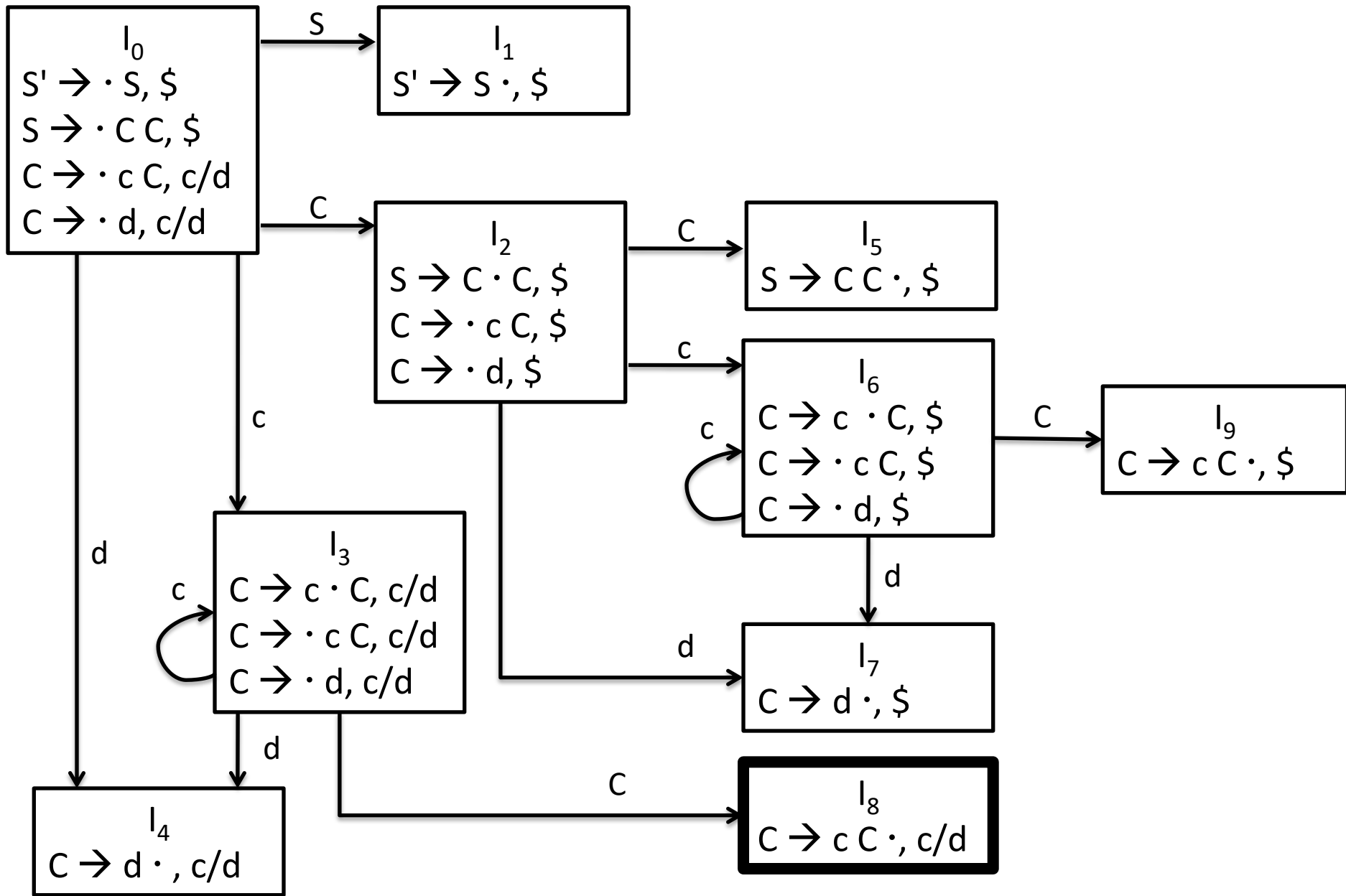


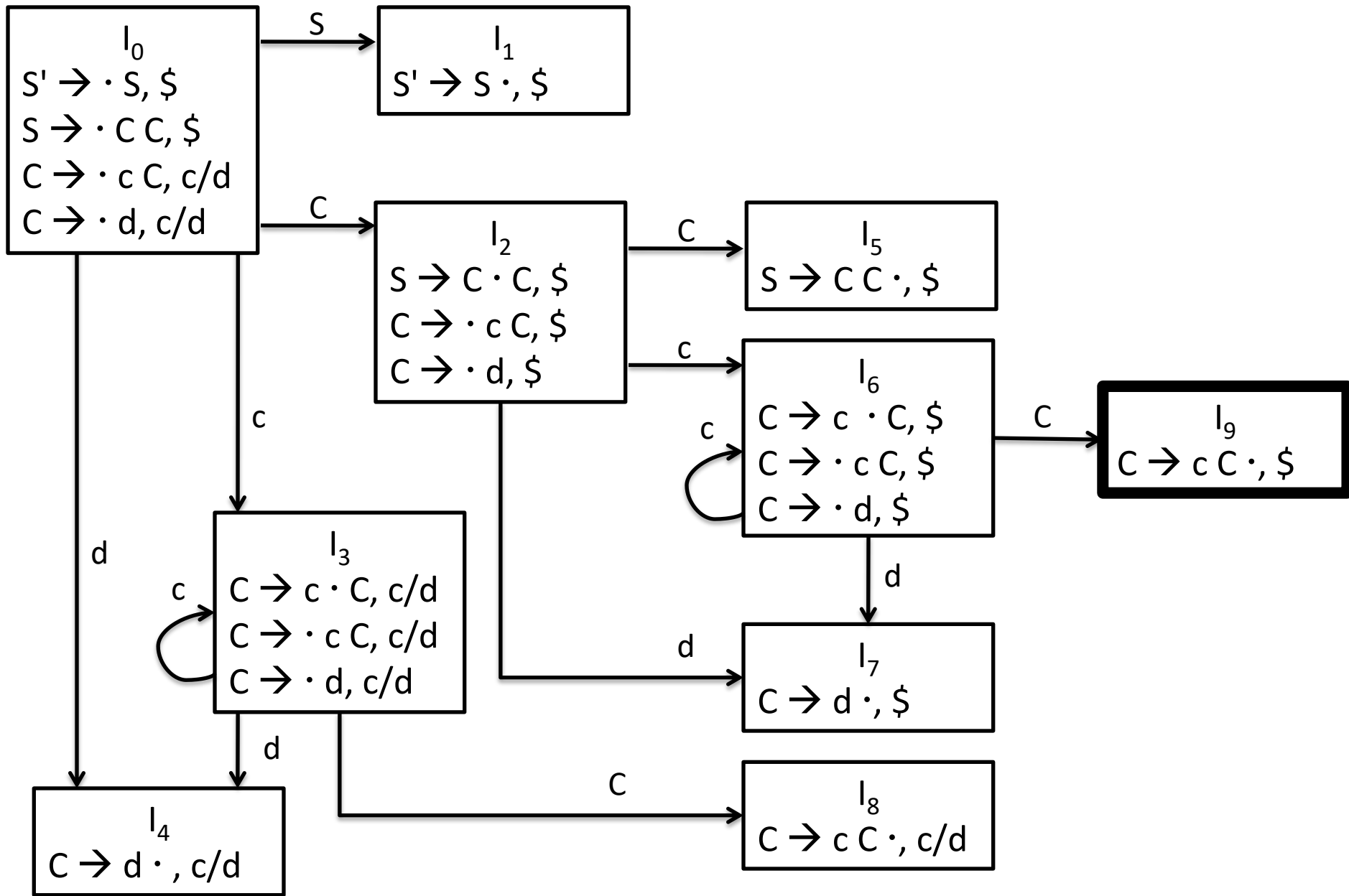












# Canonical LR(1) parsing table

| STATE | ACTION |    |    | GOTO |   |
|-------|--------|----|----|------|---|
|       | c      | d  | \$ | S    | C |
| 0     | s3     | s4 |    | 1    | 2 |
| 1     |        |    |    |      |   |
| 2     |        |    |    |      |   |
| 3     |        |    |    |      |   |
| 4     |        |    |    |      |   |
| 5     |        |    |    |      |   |
| 6     |        |    |    |      |   |
| 7     |        |    |    |      |   |
| 8     |        |    |    |      |   |
| 9     |        |    |    |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     |        |    |     |      |   |
| 3     |        |    |     |      |   |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     |        |    |     |      |   |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |



# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     |        |    |     |      |   |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     |        |    |     |      |   |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     |        |    |     |      |   |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     |        |    |     |      |   |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     |        |    |     |      |   |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

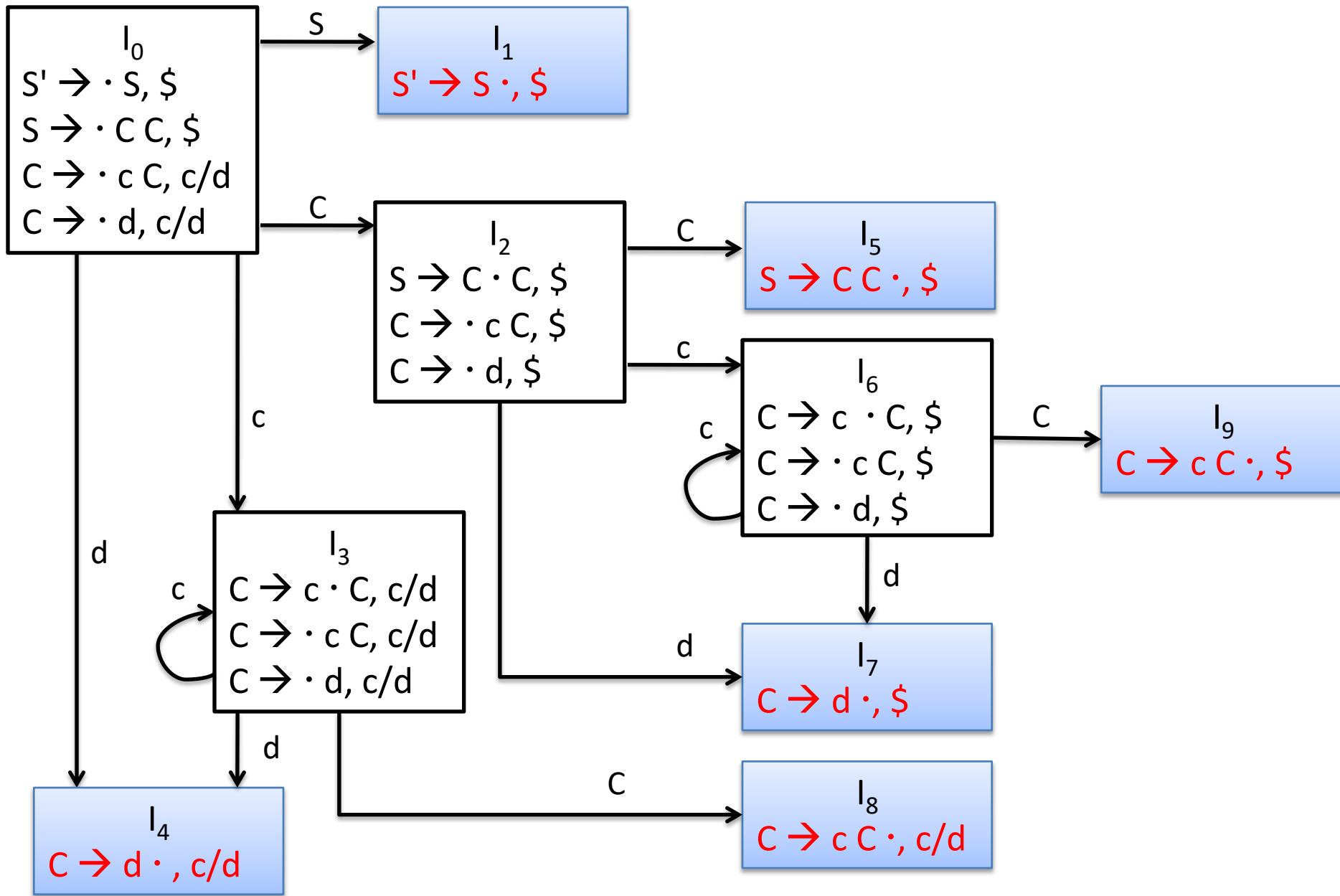
# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |





# Canonical LR(1) parsing table

- 1.  $S \rightarrow CC$
- 2.  $C \rightarrow cC$
- 3.  $C \rightarrow d$

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

$$I_1$$

$$S' \rightarrow S \cdot, \$$$

1.  $S \rightarrow c C$
2.  $C \rightarrow c C$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     |        |    |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

$I_4$   
 $C \rightarrow d \cdot, c/d$

1.  $S \rightarrow CC$
2.  $C \rightarrow cC$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     | r3     | r3 |     |      |   |
| 5     |        |    |     |      |   |
| 6     | s6     | s7 |     |      | 9 |
| 7     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

$$I_5$$

$$S \rightarrow CC; \$$$

1.  $S \rightarrow CC$
2.  $C \rightarrow cC$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | 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     |        |    |     |      |   |
| 8     |        |    |     |      |   |
| 9     |        |    |     |      |   |

$I_7$   
 $C \rightarrow d \cdot, \$$

1.  $S \rightarrow c C$
2.  $C \rightarrow c C$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | 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     |        |    |     |      |   |
| 9     |        |    |     |      |   |

$$I_8$$

$$C \rightarrow c C \cdot, c/d$$

1.  $S \rightarrow C C$
2.  $C \rightarrow c C$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | 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     |        |    |     |      |   |

$I_9$   
 $C \rightarrow c C \cdot, \$$

1.  $S \rightarrow C C$
2.  $C \rightarrow c C$
3.  $C \rightarrow d$

# Canonical LR(1) parsing table

| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | 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  |      |   |



# Canonical LR(1) parsing table

1.  $S \rightarrow CC$
2.  $C \rightarrow cC$
3.  $C \rightarrow d$

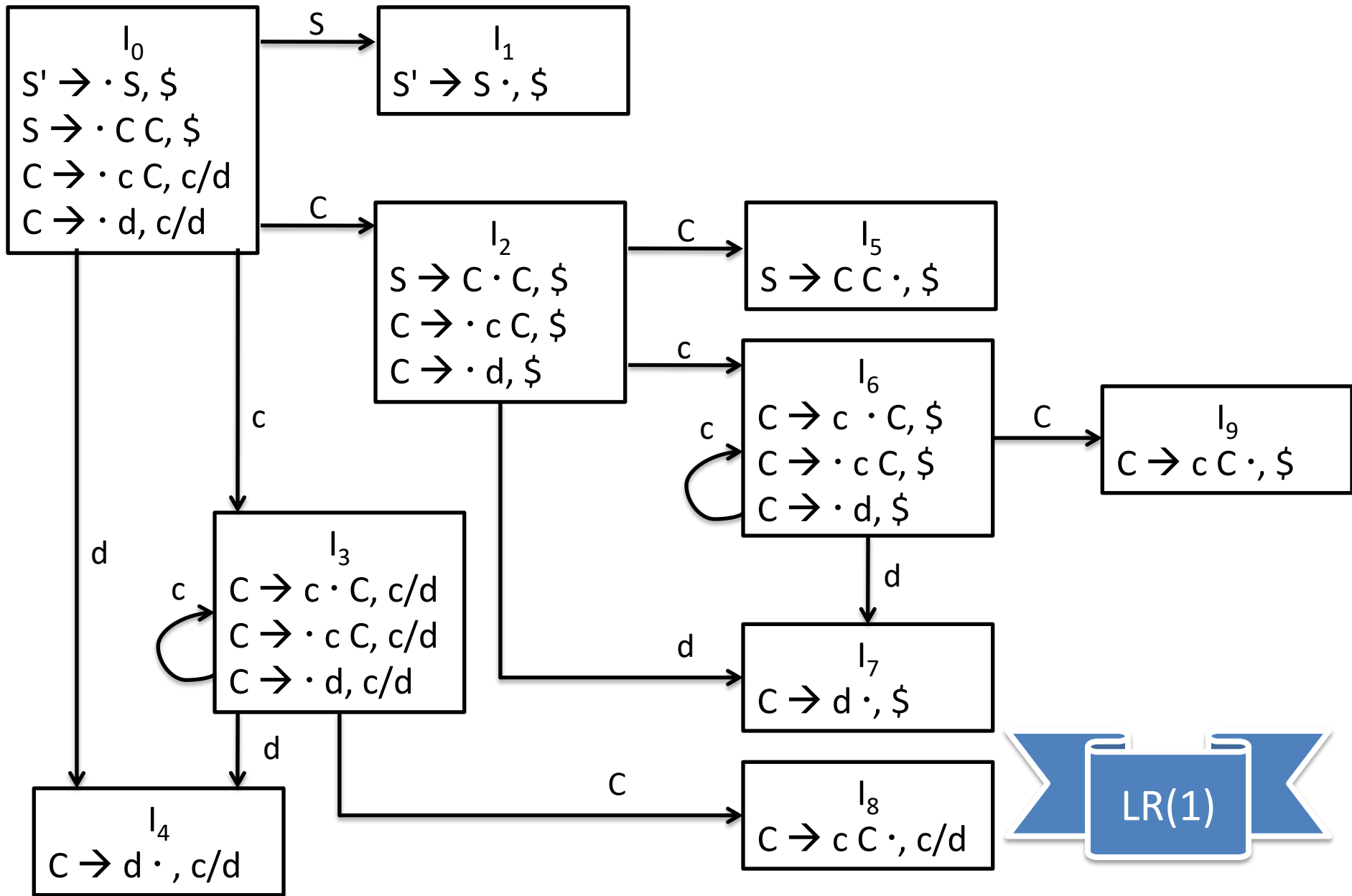
| STATE | ACTION |    |     | GOTO |   |
|-------|--------|----|-----|------|---|
|       | c      | d  | \$  | S    | C |
| 0     | s3     | s4 |     | 1    | 2 |
| 1     |        |    | acc |      |   |
| 2     | s6     | s7 |     |      | 5 |
| 3     | s3     | s4 |     |      | 8 |
| 4     | r3 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

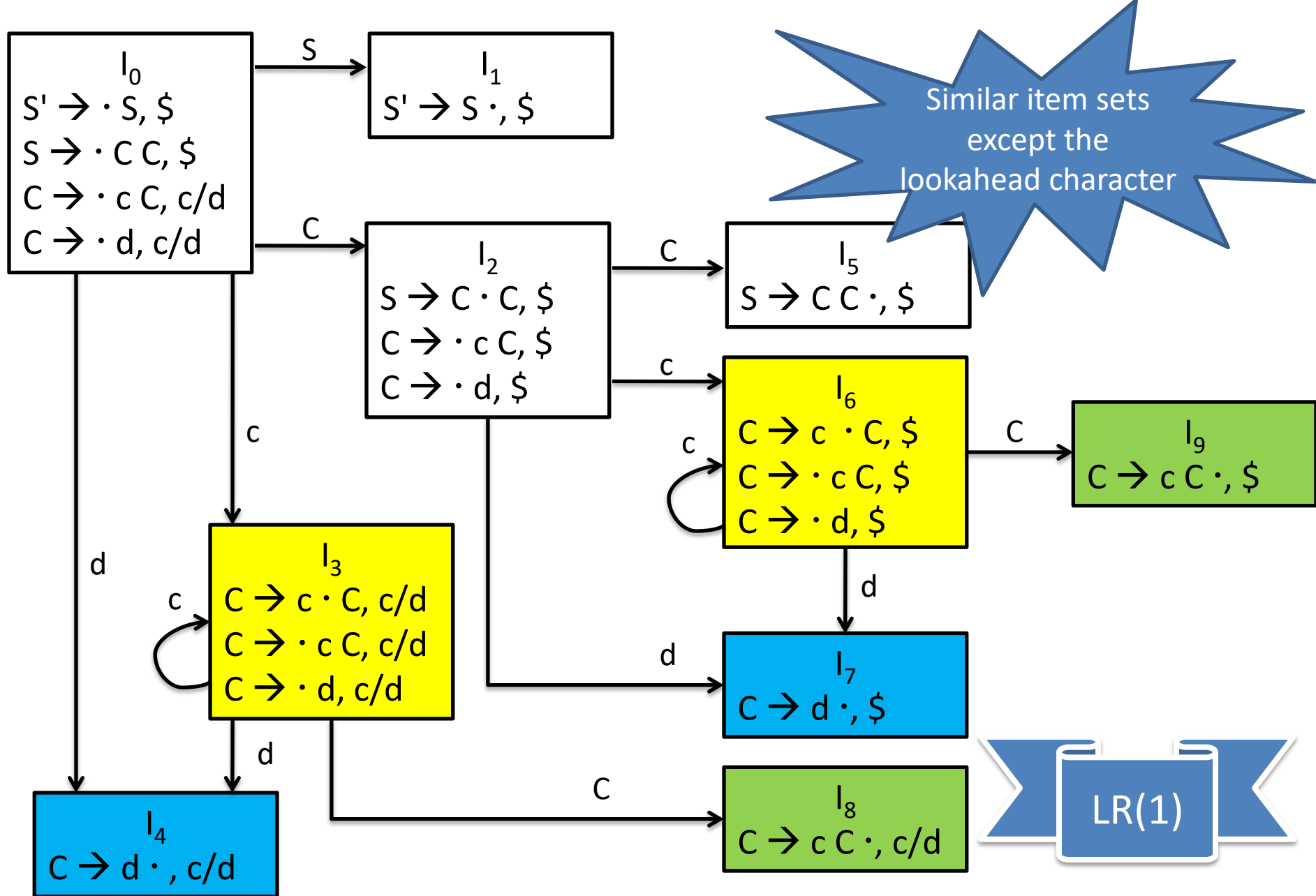
# Notes

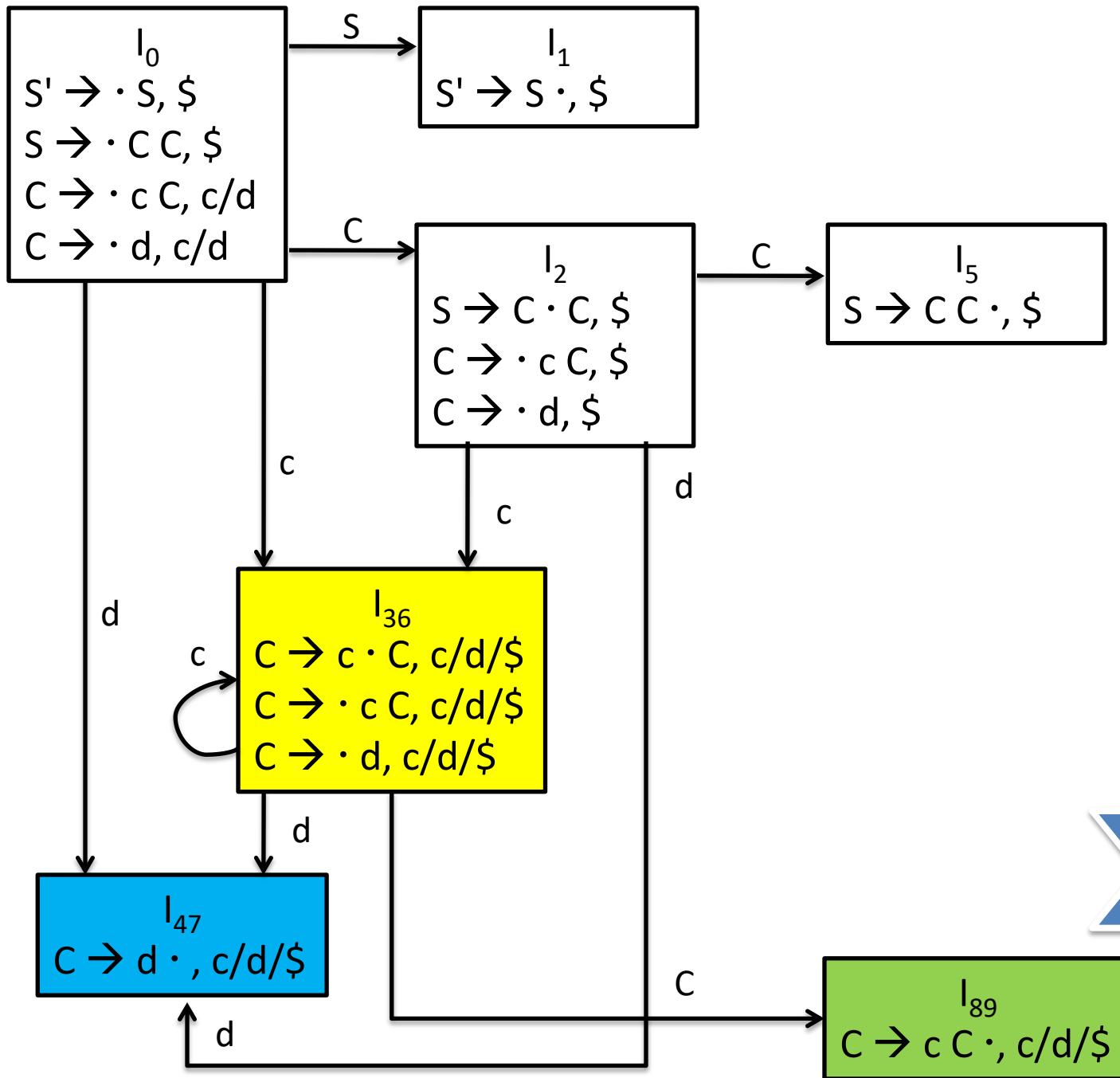
- The table formed from the parsing action and goto functions is called **the canonical LR(1) parsing table**.
- An LR parser using this table is called a **canonical-LR(1) parser**.
- If the parsing action function has no multiply defined entries, then the given grammar is called **an LR(1) grammar**. As before, we omit the "(1)" if it is understood.

# LALR Parsing [Look-Ahead LR]

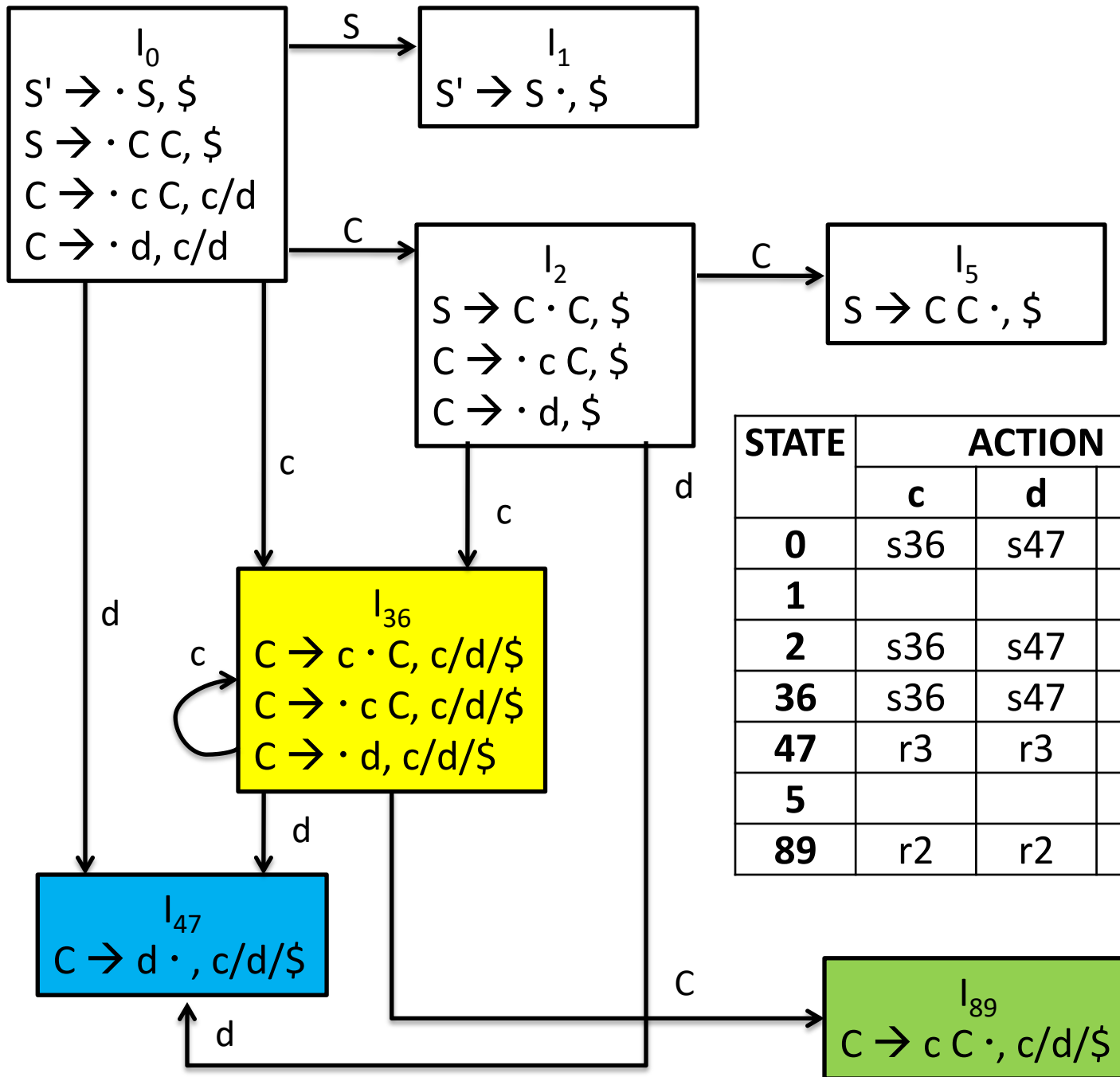
- This method is often used in practice, because the tables obtained by it are **considerably smaller** than the canonical LR tables, yet most common syntactic constructs of programming languages can be expressed conveniently by an LALR grammar.
- It is **much easier and more economical** to construct SLR and LALR tables than the canonical LR tables.







LALR(1)



| STATE     | ACTION |     |     | GOTO |    |
|-----------|--------|-----|-----|------|----|
|           | c      | d   | \$  | S    | C  |
| <b>0</b>  | s36    | s47 |     | 1    | 2  |
| <b>1</b>  |        |     | acc |      |    |
| <b>2</b>  | s36    | s47 |     |      | 5  |
| <b>36</b> | s36    | s47 |     |      | 89 |
| <b>47</b> | r3     | r3  | r3  |      |    |
| <b>5</b>  |        |     | r1  |      |    |
| <b>89</b> | r2     | r2  | r2  |      |    |

# Issue in merger

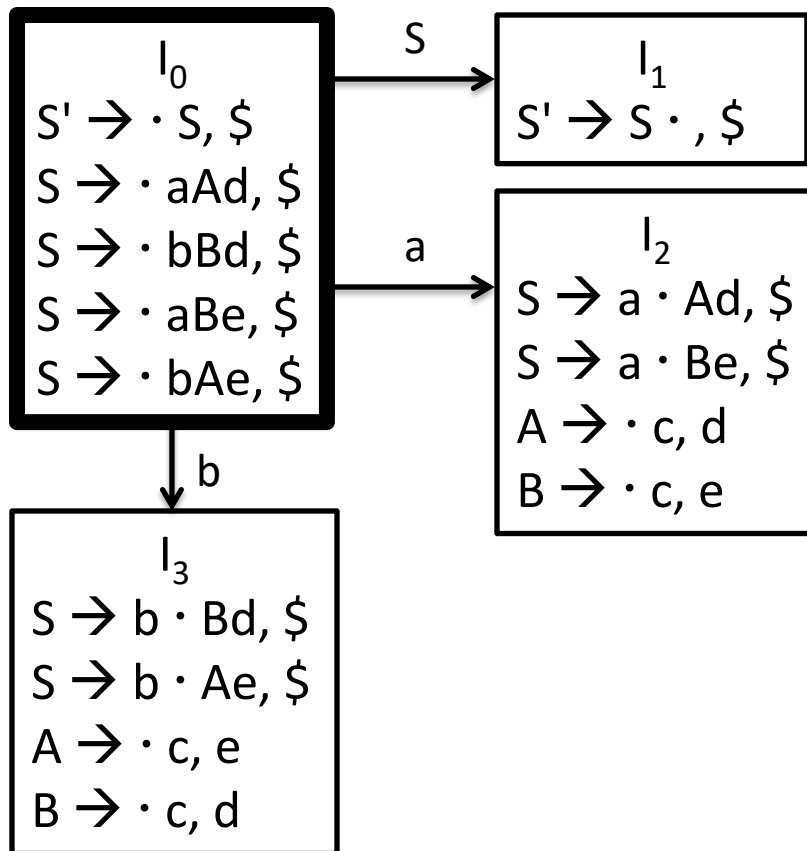
- The merging of states with common cores can never produce a **shift/reduce conflict** that was not present in one of the original states, **because shift actions depend only on the core, not the lookahead.**
- It is possible, however, that a merger will produce a **reduce/reduce conflict.**

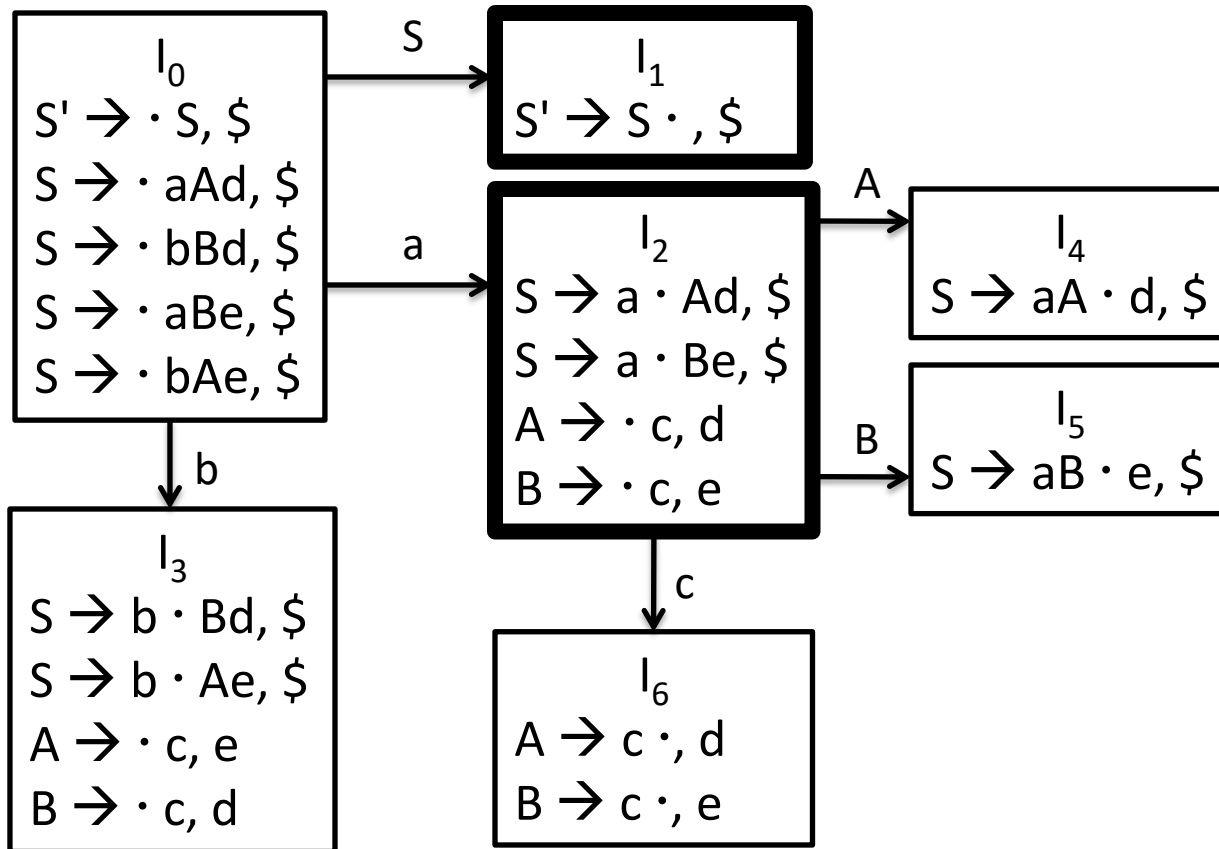


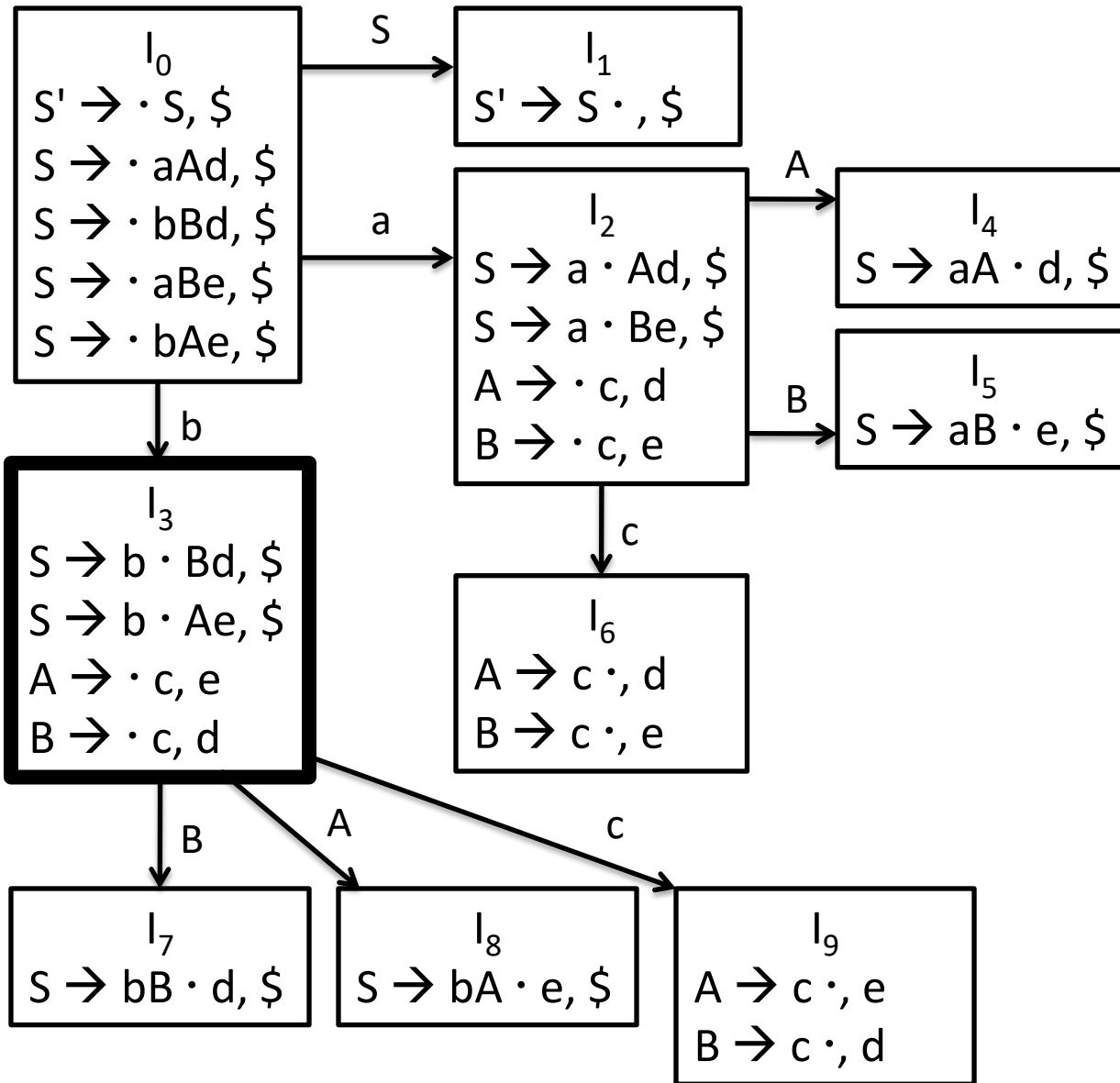
# Example to demonstrate reduce/reduce conflict after merger

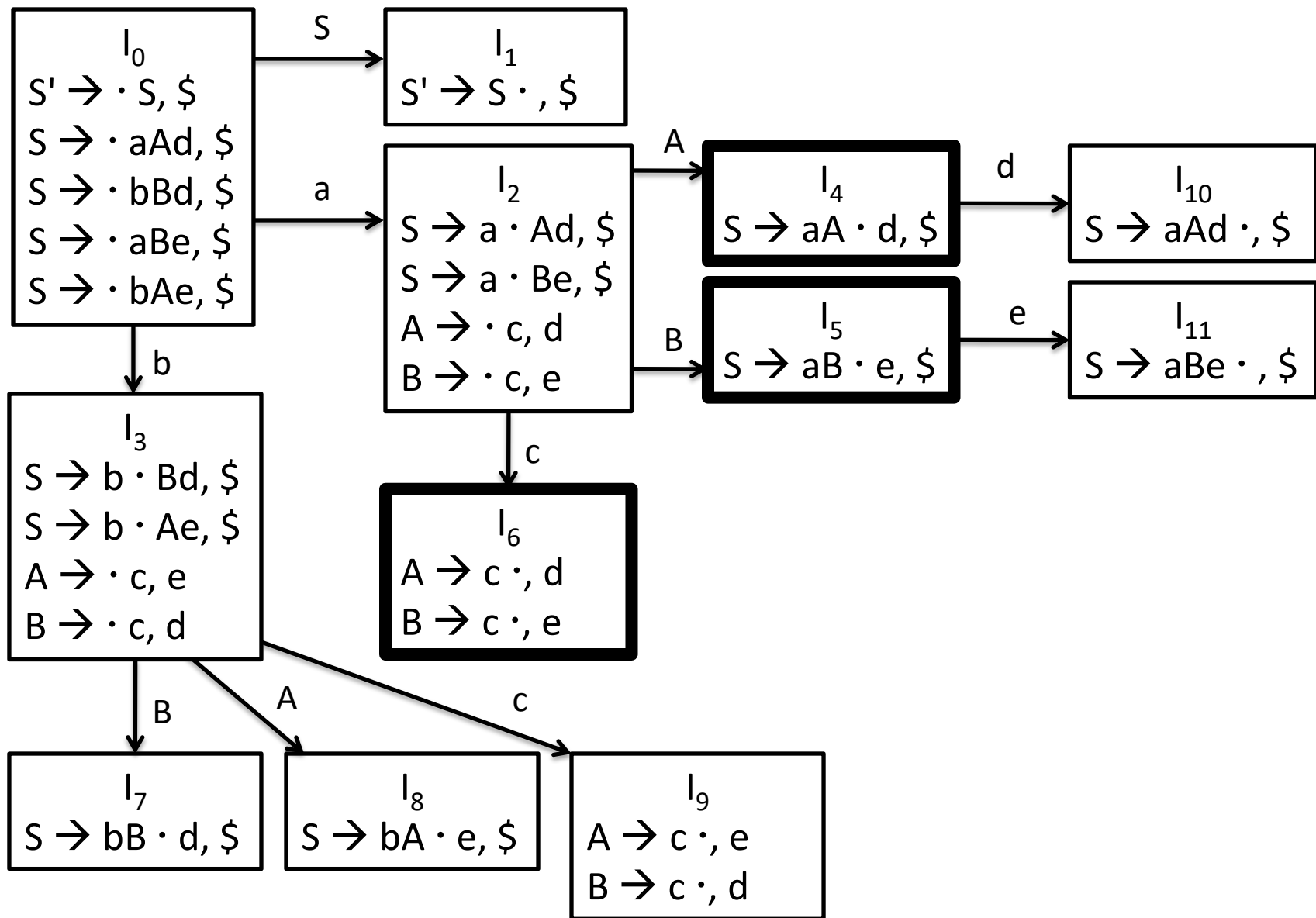
- $S' \rightarrow S$   
 $S \rightarrow aAd \mid bBd \mid aBe \mid bAe$   
 $A \rightarrow c$   
 $B \rightarrow c$

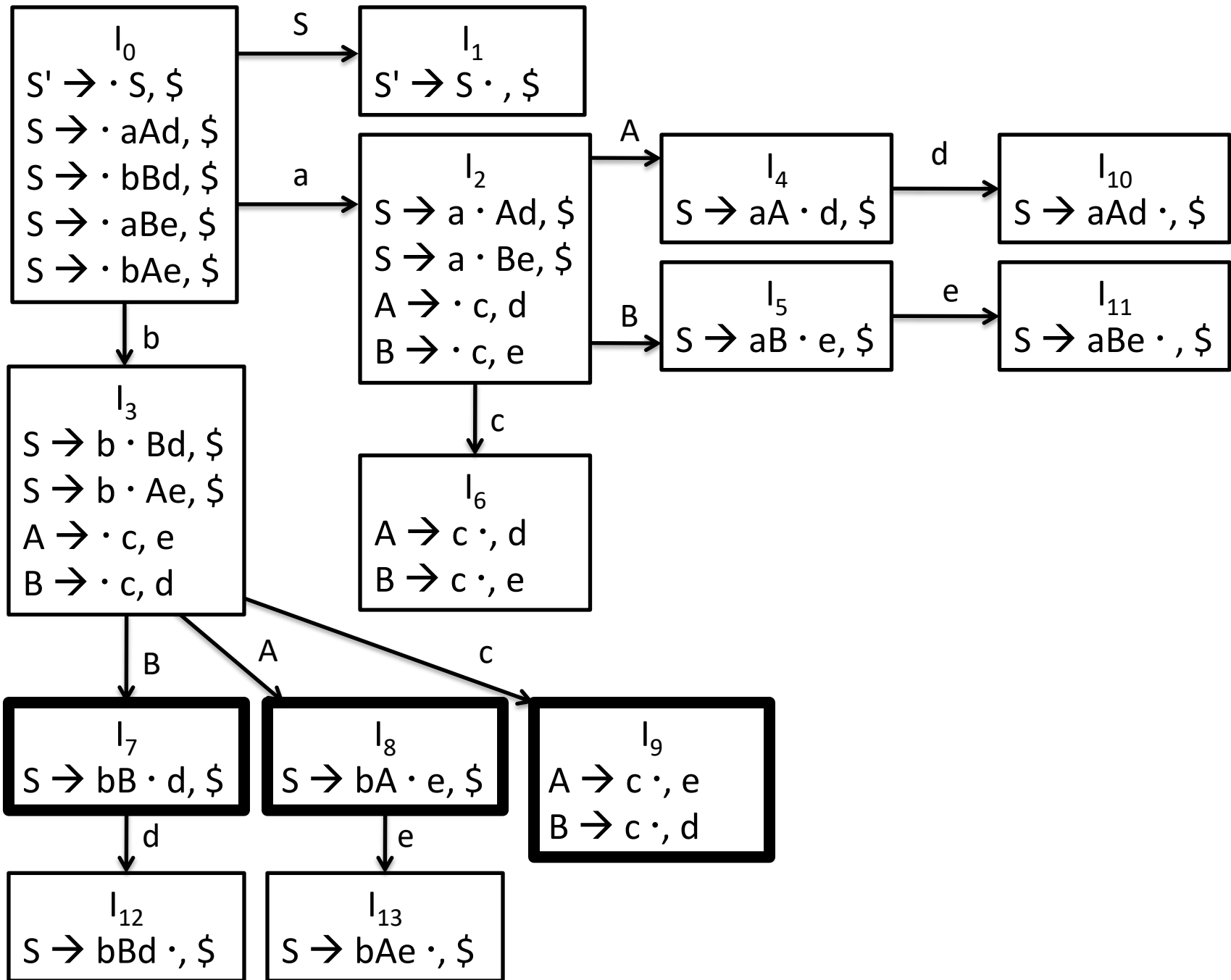
|                               |
|-------------------------------|
| $S' \rightarrow \cdot S, \$$  |
| $S \rightarrow \cdot aAd, \$$ |
| $S \rightarrow \cdot bBd, \$$ |
| $S \rightarrow \cdot aBe, \$$ |
| $S \rightarrow \cdot bAe, \$$ |

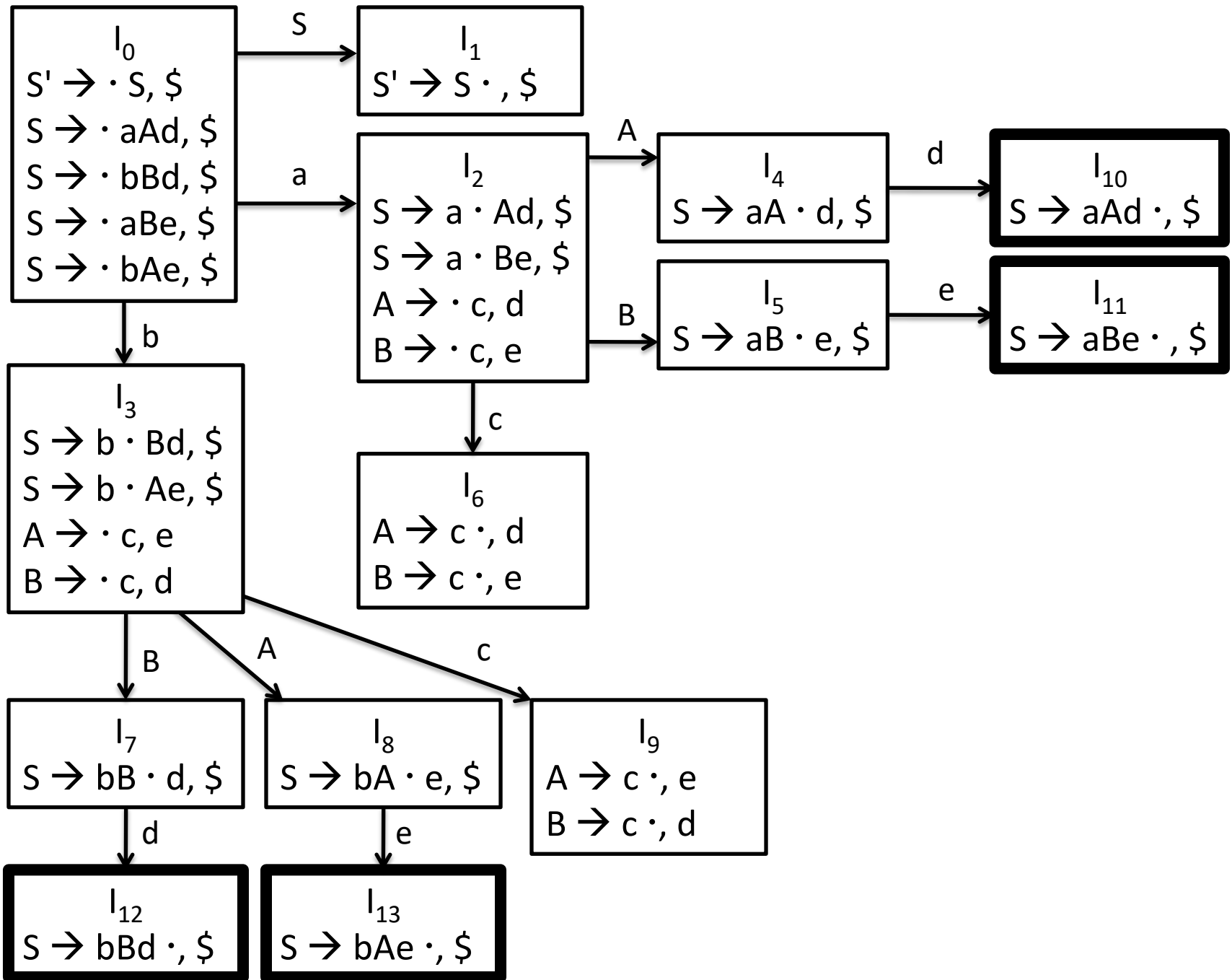


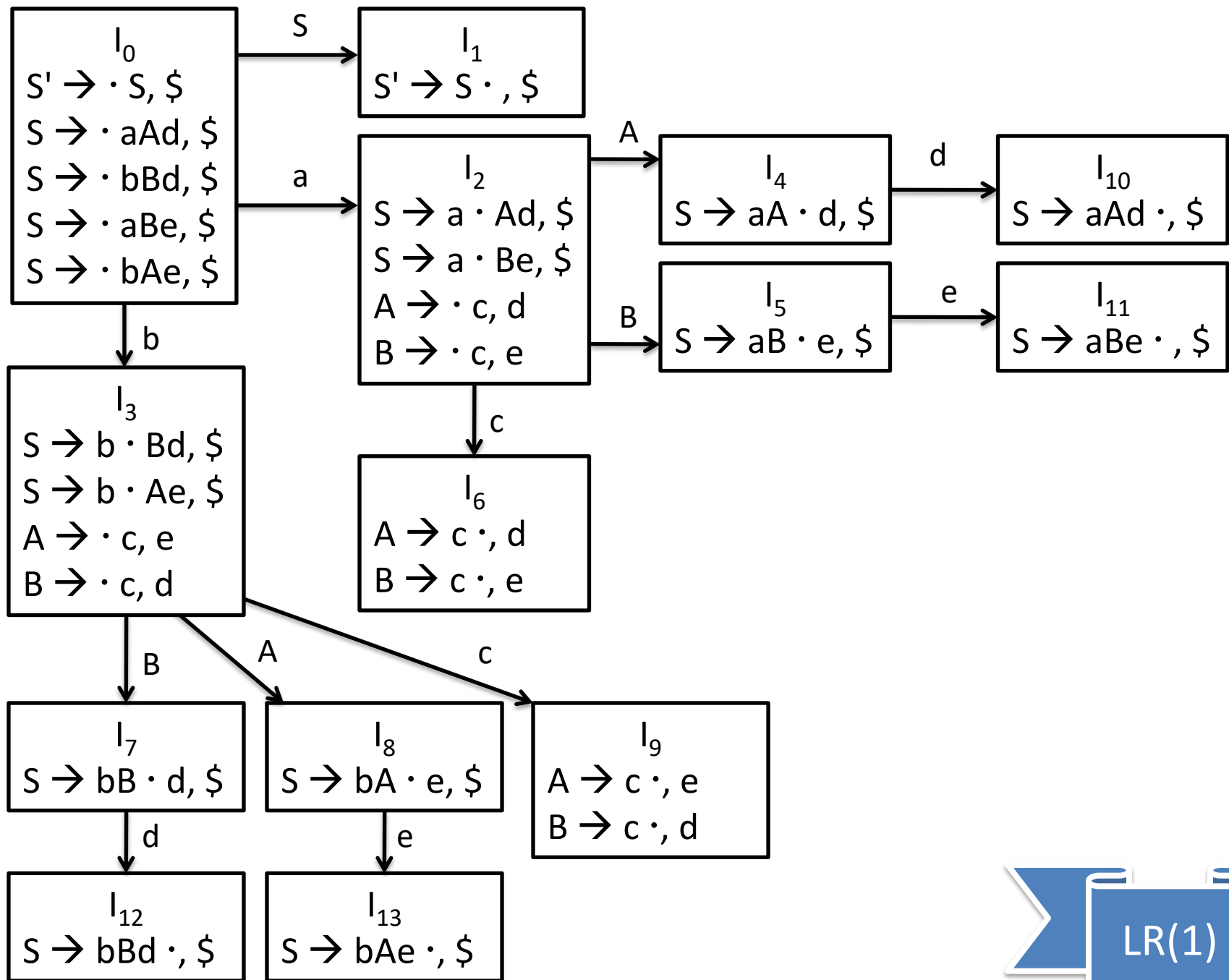




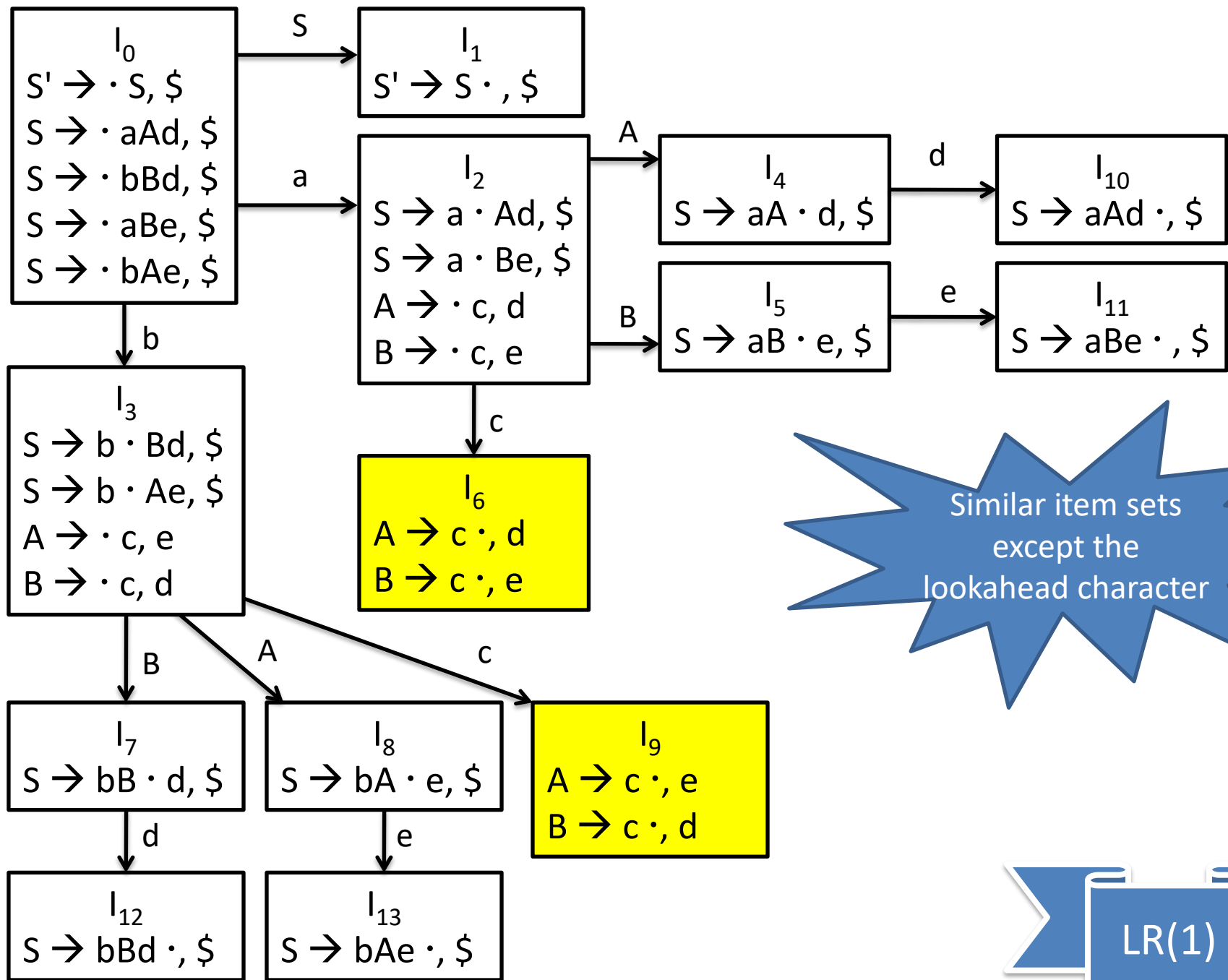






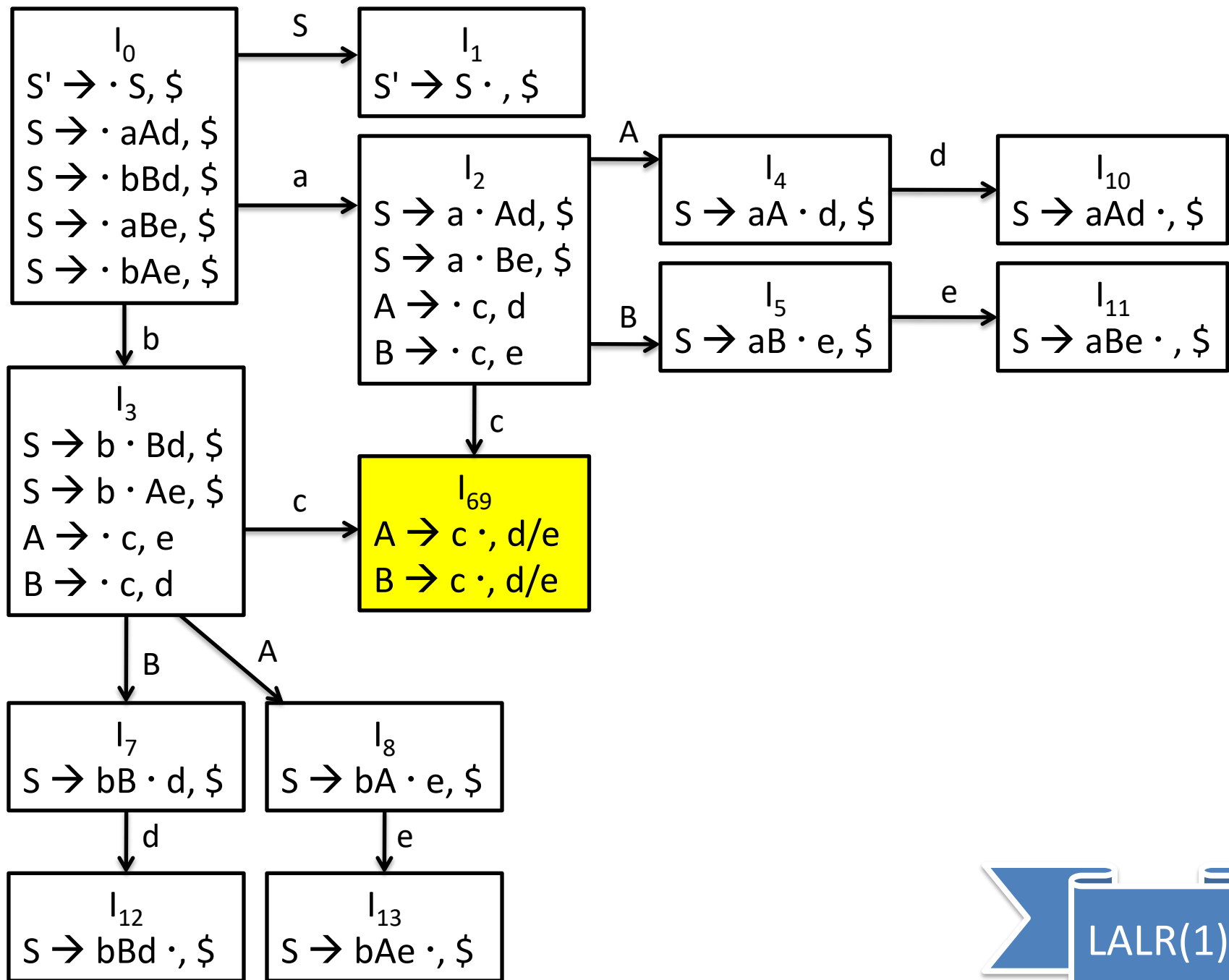


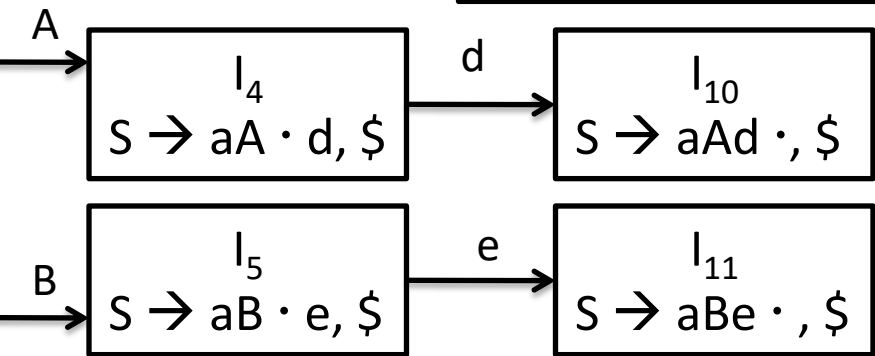
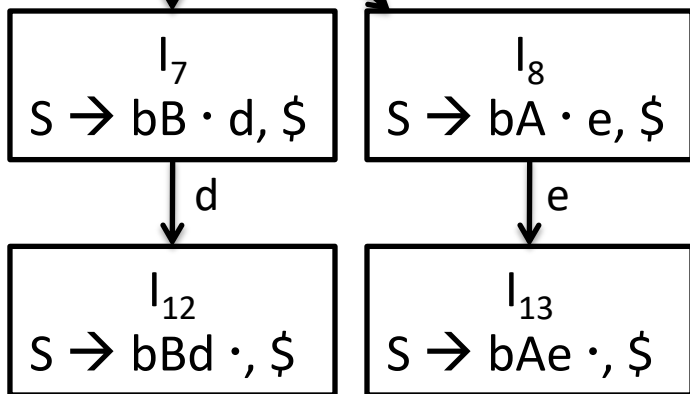
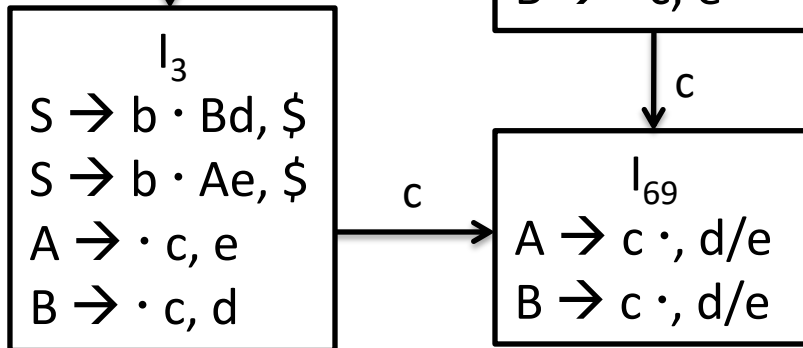
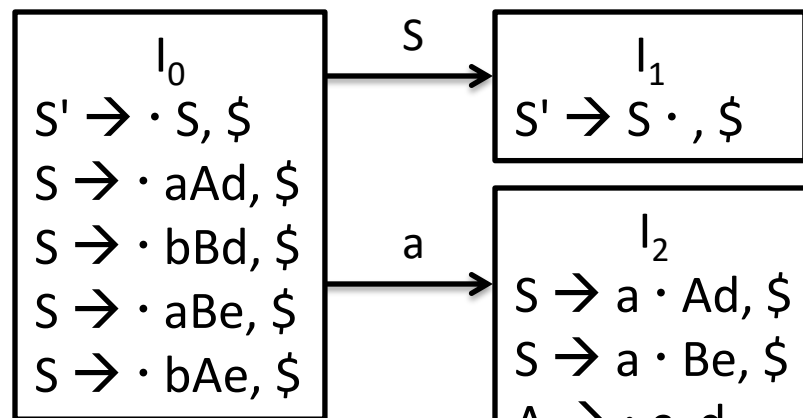




Similar item sets  
except the  
lookahead character

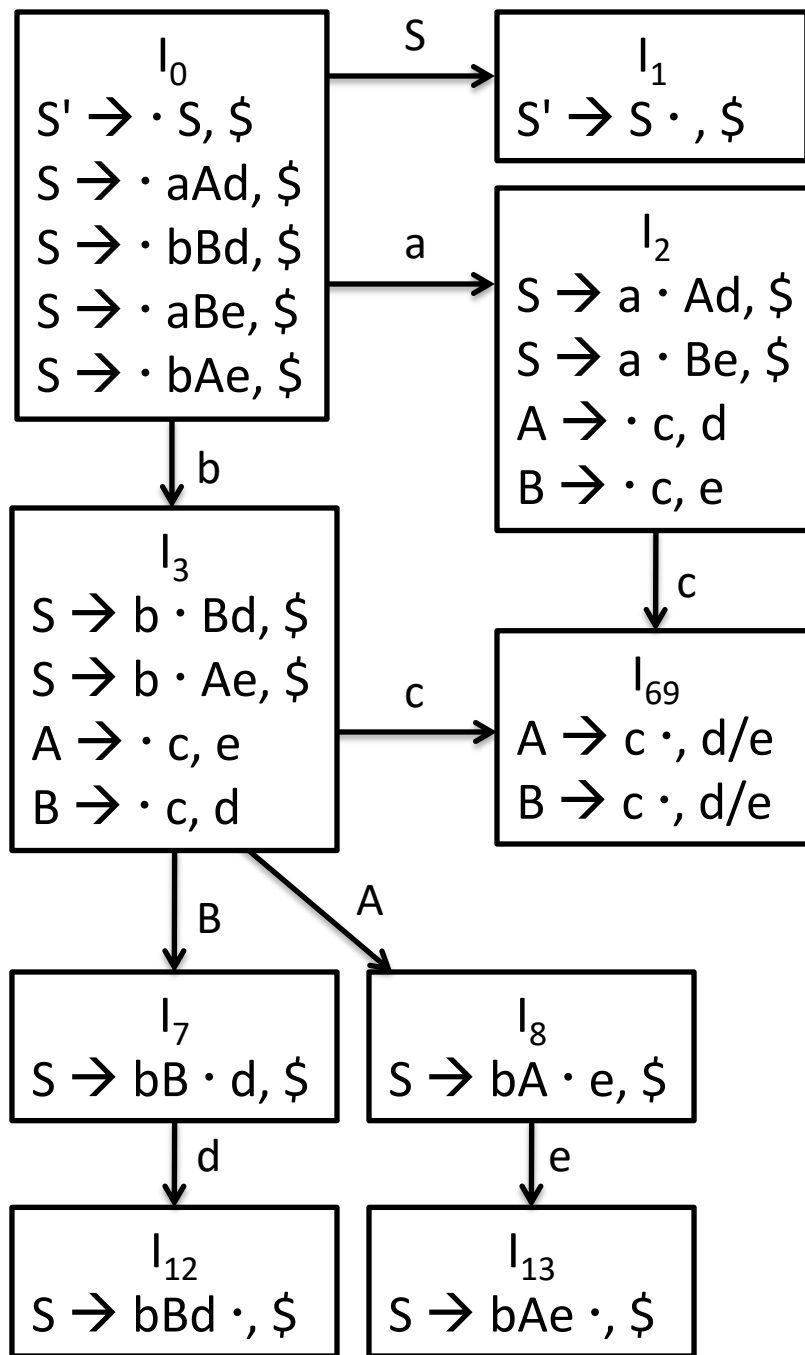
LR(1)





1.  $S \rightarrow aAd$
2.  $S \rightarrow bBd$
3.  $S \rightarrow aBe$
4.  $S \rightarrow bAe$
5.  $A \rightarrow c$
6.  $B \rightarrow c$

| STATE | ACTION |    |     |       |       |     | GOTO |   |   |
|-------|--------|----|-----|-------|-------|-----|------|---|---|
|       | a      | b  | c   | d     | e     | \$  | S    | A | B |
| 0     | s2     | s3 |     |       |       |     | 1    |   |   |
| 1     |        |    |     |       |       | acc |      |   |   |
| 2     |        |    | s69 |       |       |     |      | 4 | 5 |
| 3     |        |    | s69 |       |       |     |      | 8 | 7 |
| 4     |        |    |     | s10   |       |     |      |   |   |
| 5     |        |    |     |       | s11   |     |      |   |   |
| 69    |        |    |     | r5/r6 | r5/r6 |     |      |   |   |
| 7     |        |    |     | s12   |       |     |      |   |   |
| 8     |        |    |     |       | s13   |     |      |   |   |
| 10    |        |    |     |       |       | r1  |      |   |   |
| 11    |        |    |     |       |       | r3  |      |   |   |
| 12    |        |    |     |       |       | r2  |      |   |   |
| 13    |        |    |     |       |       | r4  |      |   |   |



1.  $S \rightarrow aAd$
2.  $S \rightarrow bBd$
3.  $S \rightarrow aBe$
4.  $S \rightarrow bAe$
5.  $A \rightarrow c$
6.  $B \rightarrow c$

reduce/reduce  
conflict in  
LALR(1)  
parsing table

| STATE | ACTION |    |       |       |    |   |   |
|-------|--------|----|-------|-------|----|---|---|
|       | a      | b  | c     | \$    |    |   |   |
| 0     | s2     | s3 |       |       |    |   |   |
| 1     |        |    |       |       | cc |   |   |
| 2     |        |    | s69   |       |    | 4 | 5 |
| 3     |        |    | s69   |       |    | 8 | 7 |
| 4     |        |    |       | s10   |    |   |   |
| 5     |        |    |       | s11   |    |   |   |
| 69    |        |    | r5/r6 | r5/r6 |    |   |   |
| 7     |        |    | s12   |       |    |   |   |
| 8     |        |    |       | s13   |    |   |   |
| 10    |        |    |       |       | r1 |   |   |
| 11    |        |    |       |       | r3 |   |   |
| 12    |        |    |       |       | r2 |   |   |
| 13    |        |    |       |       | r4 |   |   |

Show that the grammar is LALR(1) but not SLR(1)

$S \rightarrow A a \mid b A c \mid d c \mid b d a$

$A \rightarrow d$

DIY

Show that the grammar is LR(1) but not LALR(1)

$S \rightarrow A a \mid b A c \mid B c \mid b B a$

$A \rightarrow d$

$B \rightarrow d$

DIY

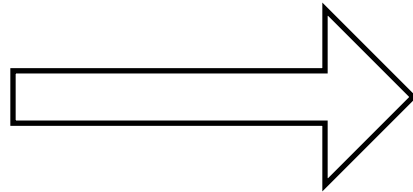
# Ambiguous Grammar

1.  $E \rightarrow E + E$

2.  $E \rightarrow E * E$

3.  $E \rightarrow (E)$

4.  $E \rightarrow id$



$E' \rightarrow \cdot E$

$E \rightarrow \cdot E + E$

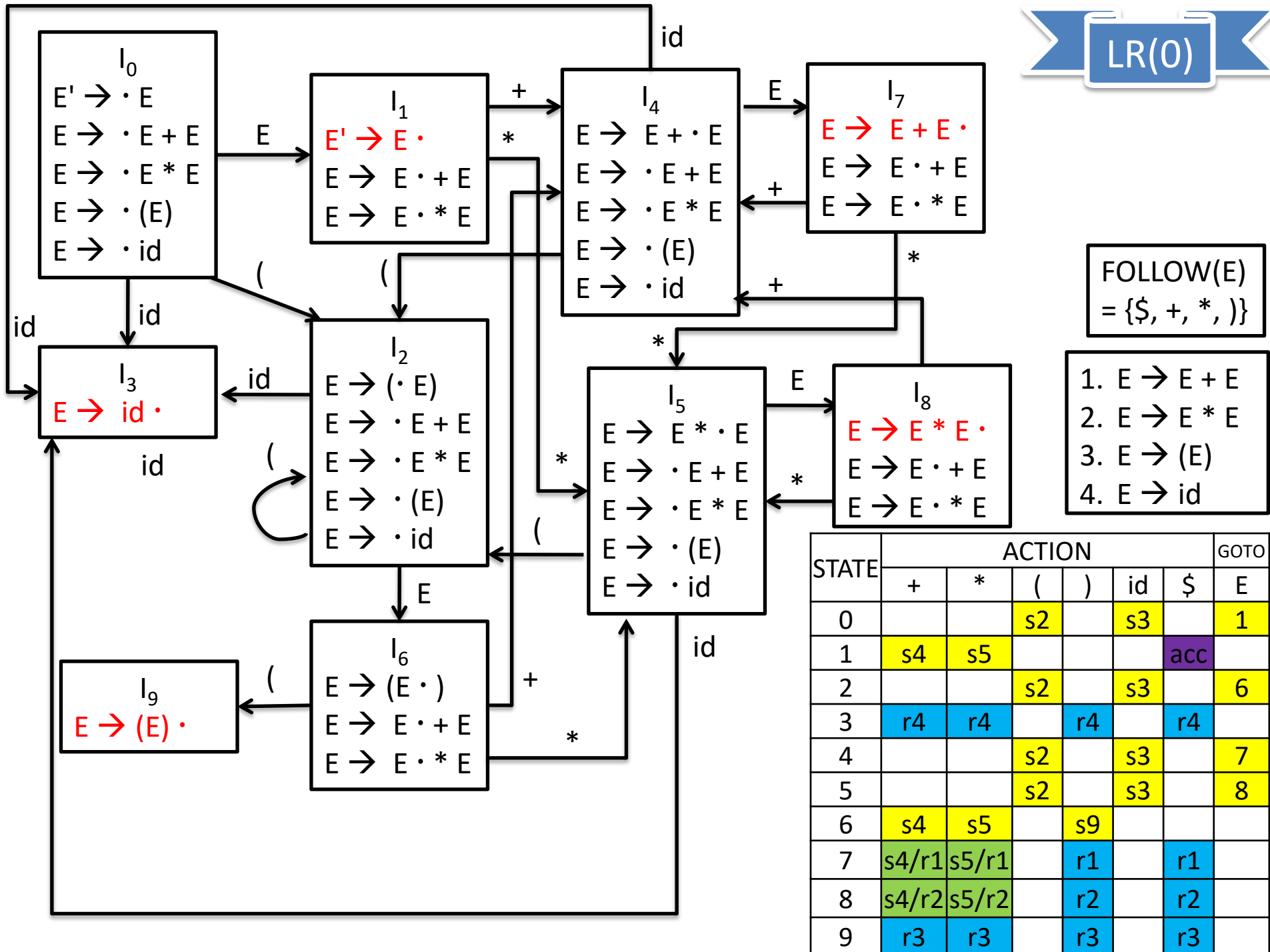
$E \rightarrow \cdot E * E$

$E \rightarrow \cdot (E)$

$E \rightarrow \cdot id$

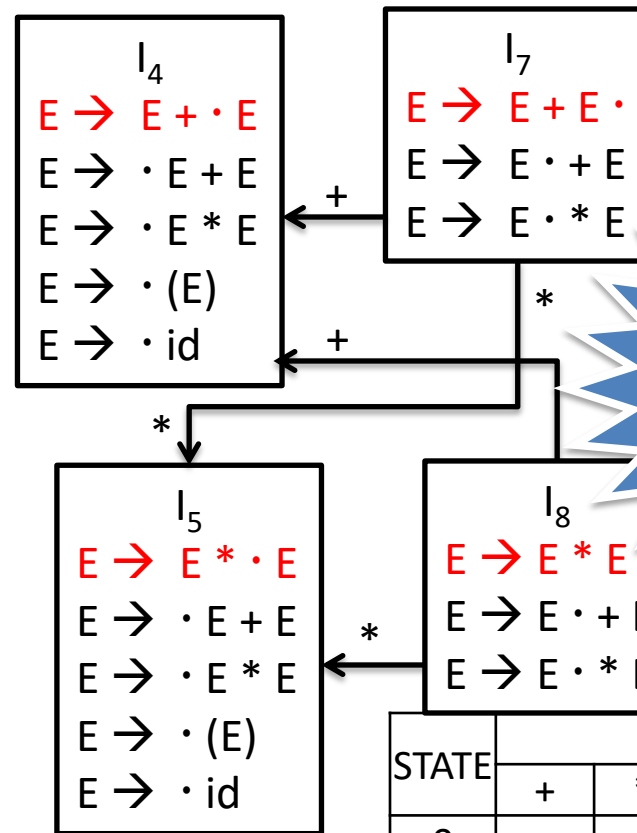
- Generate LR(0) Automaton
- Generate LR Parsing table

# LR(0)





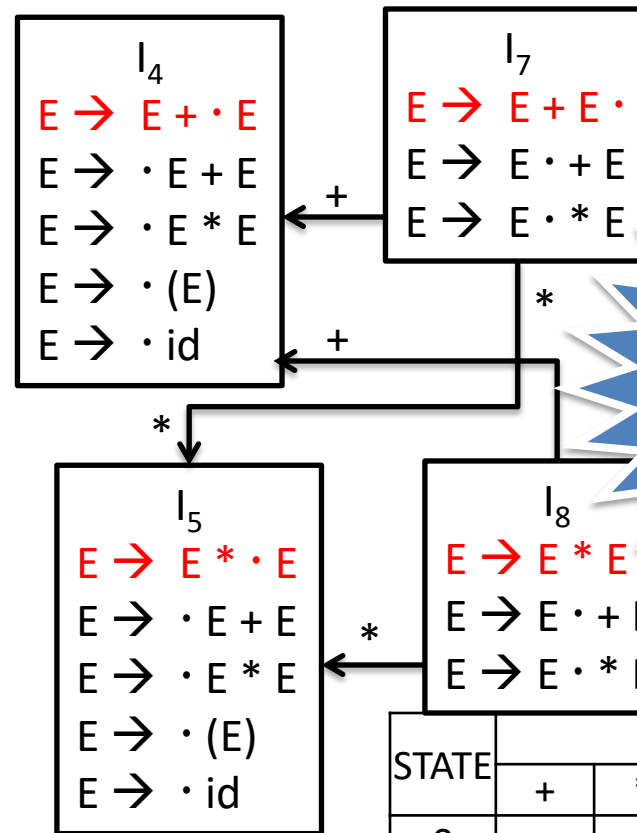
- [7, +]: + operation is performed and again the + operation to be performed.
- [7, \*]: + operation is performed and then the \* operation to be performed.
- [8, +]: \* operation is performed and then the + operation is performed.
- [8, \*]: \* operation is performed and then the \* operation is performed.



+, \* are left  
associative  
&  
\* has higher  
precedence

| STATE | ACTION |       |    |    |    |     | GOTO |
|-------|--------|-------|----|----|----|-----|------|
|       | +      | *     | (  | )  | id | \$  |      |
| 0     |        |       | s2 |    | s3 |     | 1    |
| 1     | s4     | s5    |    |    |    | acc |      |
| 2     |        |       | s2 |    | s3 |     | 6    |
| 3     | r4     | r4    |    | r4 |    | r4  |      |
| 4     |        |       | s2 |    | s3 |     | 7    |
| 5     |        |       | s2 |    | s3 |     | 8    |
| 6     | s4     | s5    |    | s9 |    |     |      |
| 7     | s4/r1  | s5/r1 |    | r1 |    | r1  |      |
| 8     | s4/r2  | s5/r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3    |    | r3 |    | r3  |      |

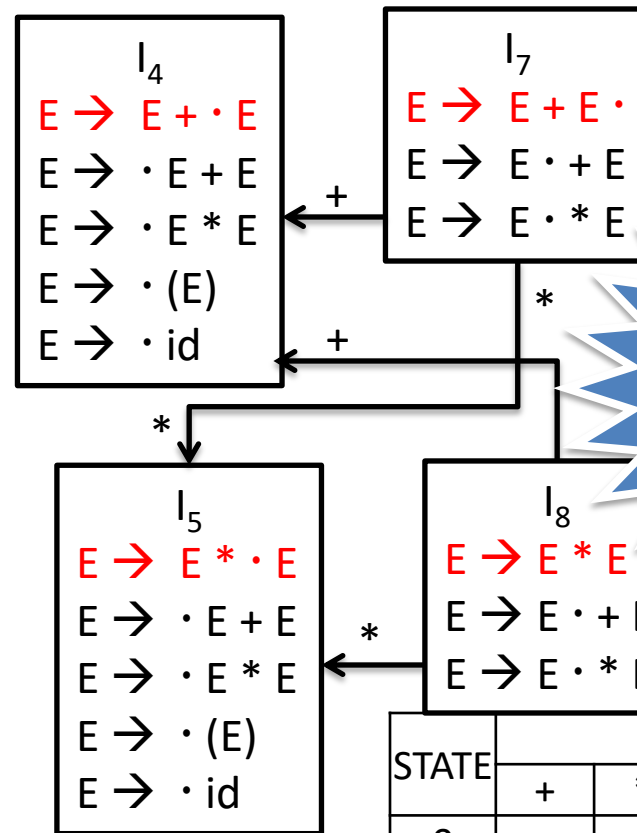
- [7, +]: + operation is performed and again the + operation to be performed. [reduce]
- [7, \*]: + operation is performed and then the \* operation to be performed.
- [8, +]: \* operation is performed and then the + operation is performed.
- [8, \*]: \* operation is performed and then the \* operation is performed. [reduce]



+, \* are left  
associative  
&  
\* has higher  
precedence

| STATE | ACTION |       |    |    |    |     | GOTO |
|-------|--------|-------|----|----|----|-----|------|
|       | +      | *     | (  | )  | id | \$  |      |
| 0     |        |       | s2 |    | s3 |     | 1    |
| 1     | s4     | s5    |    |    |    | acc |      |
| 2     |        |       | s2 |    | s3 |     | 6    |
| 3     | r4     | r4    |    | r4 |    | r4  |      |
| 4     |        |       | s2 |    | s3 |     | 7    |
| 5     |        |       | s2 |    | s3 |     | 8    |
| 6     | s4     | s5    |    | s9 |    |     |      |
| 7     | r1     | s5/r1 |    | r1 |    | r1  |      |
| 8     | s4/r2  | r2    |    | r2 |    | r2  |      |
| 9     | r3     | r3    |    | r3 |    | r3  |      |

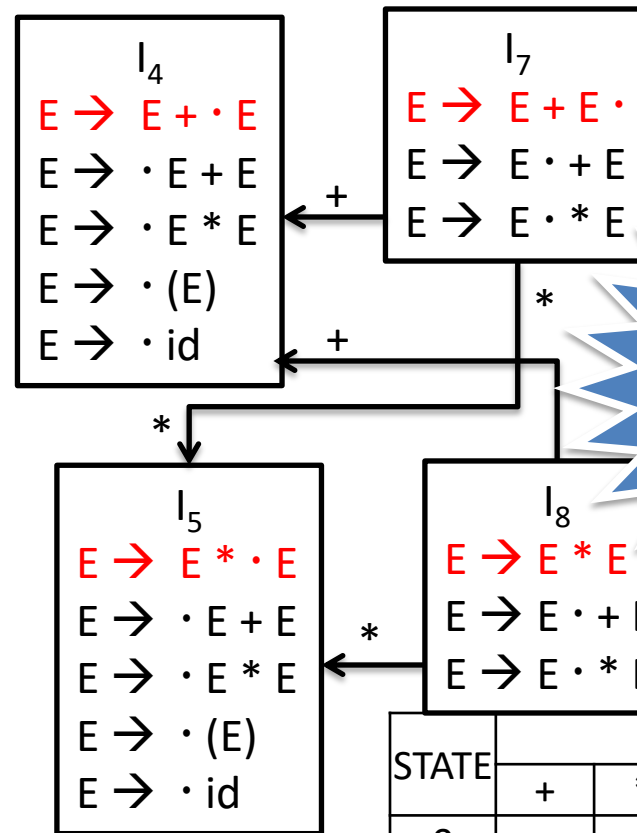
- $[7, +]$ :  $+$  operation is performed and again the  $+$  operation to be performed. **[reduce]**
- $[7, *]$ :  $+$  operation is performed and then the  $*$  operation to be performed.
- $[8, +]$ :  $*$  operation is performed and then the  $+$  operation is performed. **[reduce]**
- $[8, *]$ :  $*$  operation is performed and then the  $*$  operation is performed. **[reduce]**



$+, *$  are left  
associative  
&  
 $*$  has higher  
precedence

| STATE | ACTION |       |    |    |    |     | GOTO |
|-------|--------|-------|----|----|----|-----|------|
|       | +      | *     | (  | )  | id | \$  |      |
| 0     |        |       | s2 |    | s3 |     | 1    |
| 1     | s4     | s5    |    |    |    | acc |      |
| 2     |        |       | s2 |    | s3 |     | 6    |
| 3     | r4     | r4    |    | r4 |    | r4  |      |
| 4     |        |       | s2 |    | s3 |     | 7    |
| 5     |        |       | s2 |    | s3 |     | 8    |
| 6     | s4     | s5    |    | s9 |    |     |      |
| 7     | r1     | s5/r1 |    | r1 |    | r1  |      |
| 8     | r2     | r2    |    | r2 |    | r2  |      |
| 9     | r3     | r3    |    | r3 |    | r3  |      |

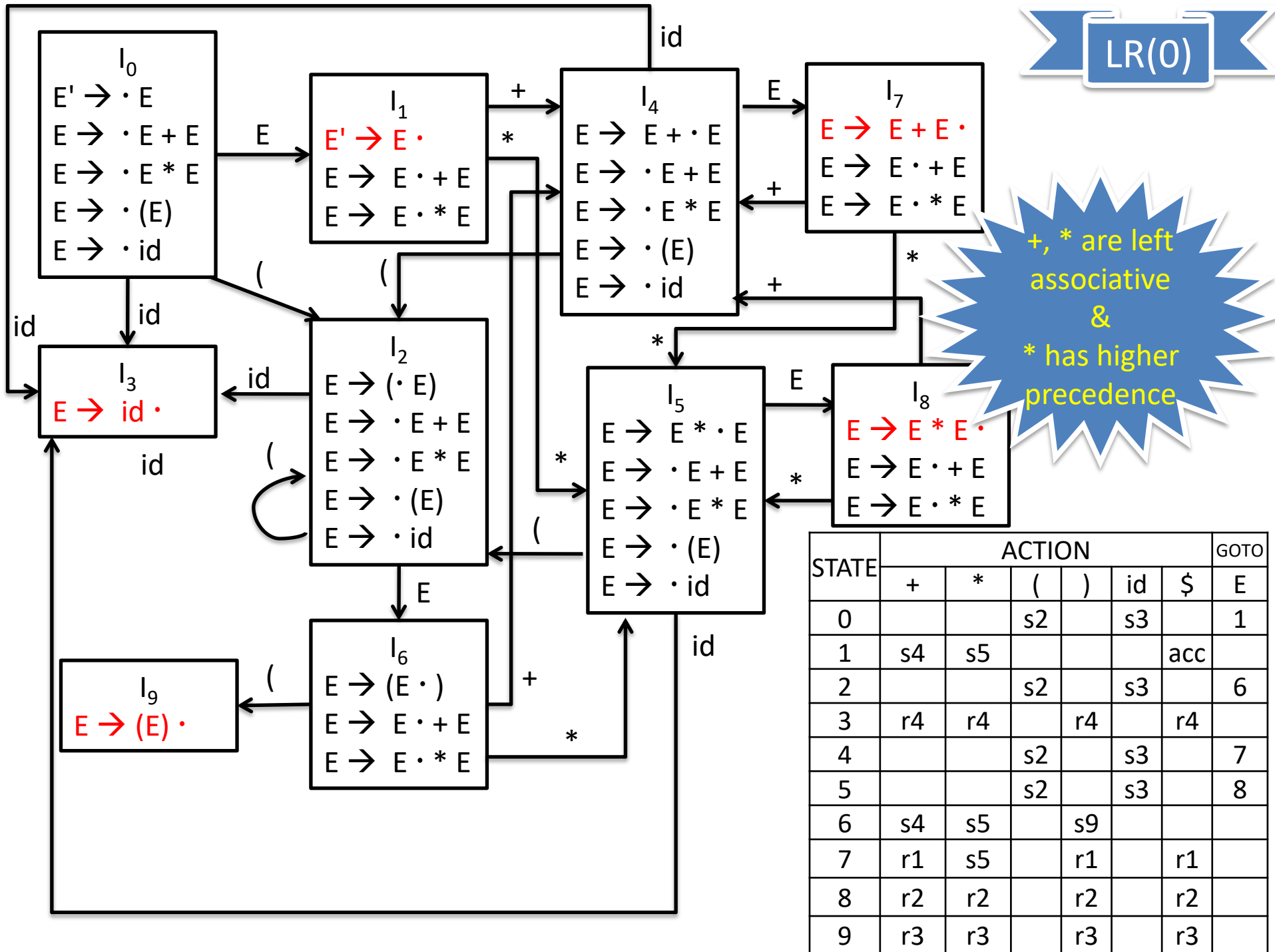
- [7, +]: + operation is performed and again the + operation to be performed. [reduce]
- [7, \*]: + operation is performed and then the \* operation to be performed.
- [8, +]: \* operation is performed and then the + operation is performed. [reduce]
- [8, \*]: \* operation is performed and then the \* operation is performed. [reduce]



+, \* are left  
associative  
&  
\* has higher  
precedence

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  |      |
| 0     |        |    | s2 |    | s3 |     | 1    |
| 1     | s4     | s5 |    |    |    | acc |      |
| 2     |        |    | s2 |    | s3 |     | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     |        |    | s2 |    | s3 |     | 7    |
| 5     |        |    | s2 |    | s3 |     | 8    |
| 6     | s4     | s5 |    | s9 |    |     |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

LR(0)



# The “Dangling-Else” Ambiguity

- Consider the grammar for conditional statements:

stmt  $\rightarrow$  if expr then stmt else stmt

    | if expr then stmt

    | other

- Augmented Grammar

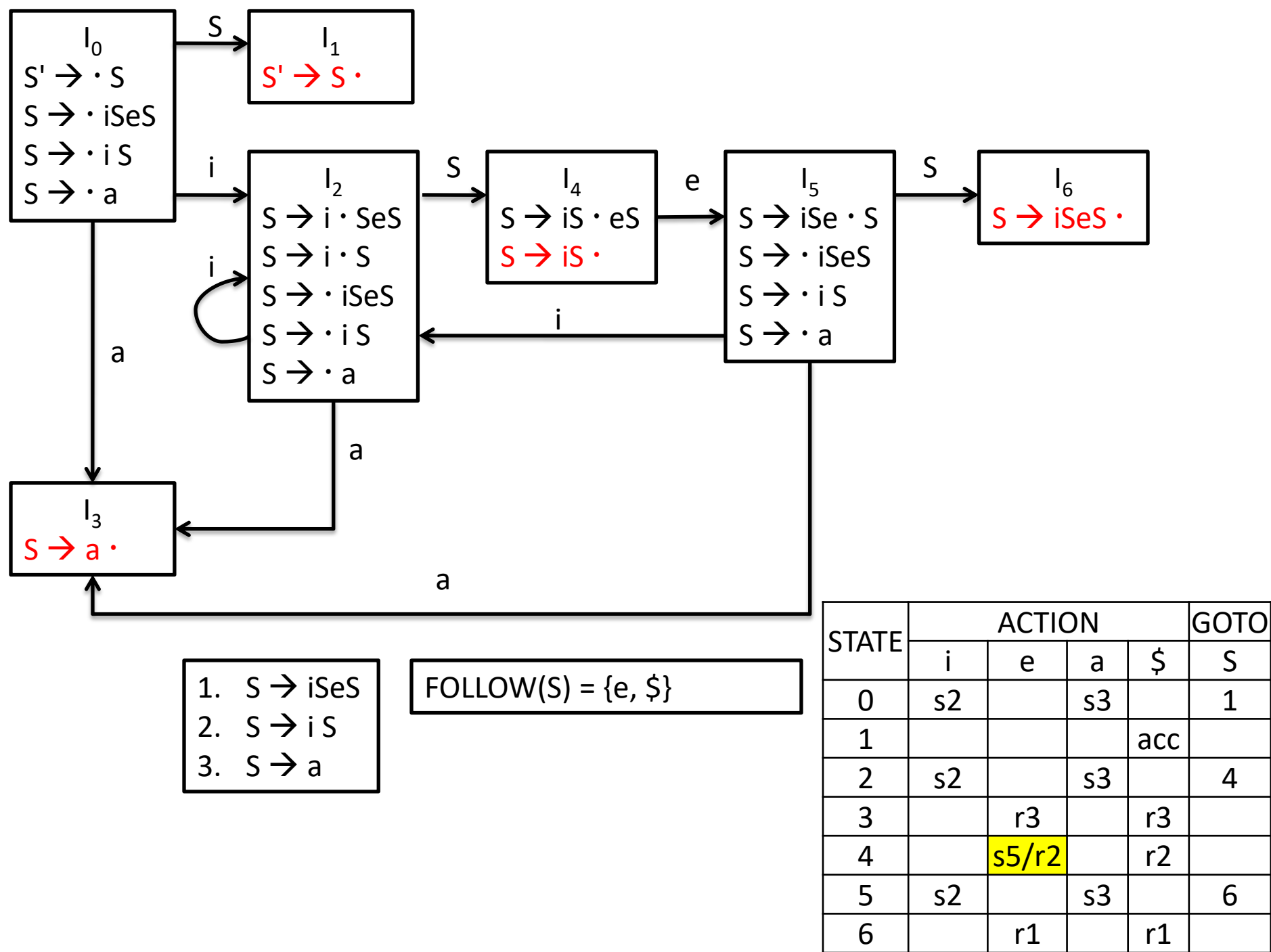
$S' \rightarrow S$

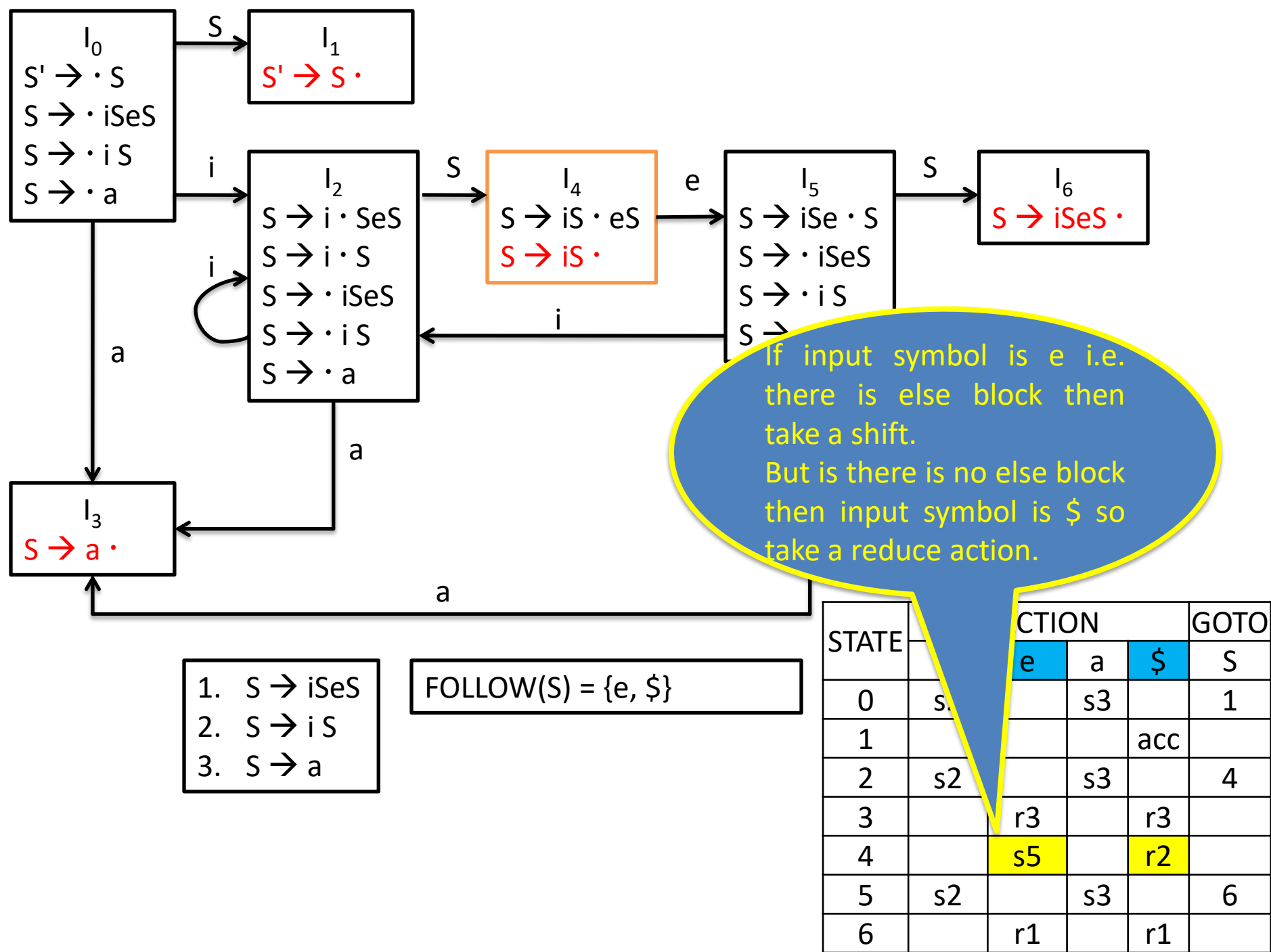
$S \rightarrow iSeS$

$S \rightarrow iS$

$S \rightarrow a$

|       |                          |
|-------|--------------------------|
| $I_0$ |                          |
| $S'$  | $\rightarrow \cdot S$    |
| $S$   | $\rightarrow \cdot iSeS$ |
| $S$   | $\rightarrow \cdot iS$   |
| $S$   | $\rightarrow \cdot a$    |







# Error Recovery

- An LR parser will detect an error when it consults the parsing action table and finds an error entry.
- Errors are never detected by consulting the goto table.
- An LR parser will announce an error as soon as there is no valid continuation for the portion of the input thus far scanned.
- A **canonical LR parser** will not make even a single reduction before announcing an error.
- **SLR and LALR parsers** may make several reductions before announcing an error, but they will never shift an erroneous input symbol onto the stack.

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  |      |
| 0     | e1     | e1 | s2 |    | s3 | e1  | 1    |
| 1     | s4     | s5 |    |    |    | acc |      |
| 2     | e1     | e1 | s2 |    | s3 | e1  | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     | e1     | e1 | s2 |    | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 |    | s3 | e1  | 8    |
| 6     | s4     | s5 |    | s9 |    |     |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

e1:

- This routine is called from states 0, 2, 4 and 5, all of which expect the beginning of an operand, either an id or a left parenthesis.
- Instead, +, \* , or the end of the input was found.
- push state 3 (the goto of states 0, 2, 4 and 5 on id);
- Issue diagnostic “missing operand”

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  |      |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 |    | e2 |    | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 |    | s9 |    |     |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

e2:

- Called from states 0, 1, 2, 4 and 5 on finding a right parenthesis
- remove the right parenthesis from the input;
- issue diagnostic “unbalanced right parenthesis”.

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  |      |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 | e3 | e2 | e3 | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 | e3 | s9 | e3 |     |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

e3:

- Called from states 1 or 6 when expecting an operator, and an id or left parenthesis is found.
- push state 4 (corresponding to symbol + ) onto the stack;
- push state 5 (corresponding to symbol \* ) onto the stack;
- issue diagnostic “missing operator”.

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  | E    |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 | e3 | e2 | e3 | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 | e3 | s9 | e3 | e4  |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

e4:

- Called from state 6 when the end of the input is found.
- push state 9 (for a right parenthesis) onto the stack;
- issue diagnostic “missing right parenthesis”.

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  | E    |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 | e3 | e2 | e3 | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 |    | r4 |    | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 | e3 | s9 | e3 | e4  |      |
| 7     | r1     | s5 |    | r1 |    | r1  |      |
| 8     | r2     | r2 |    | r2 |    | r2  |      |
| 9     | r3     | r3 |    | r3 |    | r3  |      |

All other cells are filled with  
**respective reduce entries.**

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  |      |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 | e3 | e2 | e3 | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 | r4 | r4 | r4 | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 | e3 | s9 | e3 | e4  |      |
| 7     | r1     | s5 | r1 | r1 | r1 | r1  |      |
| 8     | r2     | r2 | r2 | r2 | r2 | r2  |      |
| 9     | r3     | r3 | r3 | r3 | r3 | r3  |      |

All other cells are filled with  
**respective reduce entries.**

# Error Routines

| STATE | ACTION |    |    |    |    |     | GOTO |
|-------|--------|----|----|----|----|-----|------|
|       | +      | *  | (  | )  | id | \$  | E    |
| 0     | e1     | e1 | s2 | e2 | s3 | e1  | 1    |
| 1     | s4     | s5 | e3 | e2 | e3 | acc |      |
| 2     | e1     | e1 | s2 | e2 | s3 | e1  | 6    |
| 3     | r4     | r4 | r4 | r4 | r4 | r4  |      |
| 4     | e1     | e1 | s2 | e2 | s3 | e1  | 7    |
| 5     | e1     | e1 | s2 | e2 | s3 | e1  | 8    |
| 6     | s4     | s5 | e3 | s9 | e3 | e4  |      |
| 7     | r1     | s5 | r1 | r1 | r1 | r1  |      |
| 8     | r2     | r2 | r2 | r2 | r2 | r2  |      |
| 9     | r3     | r3 | r3 | r3 | r3 | r3  |      |