Compiler Design

Fatemeh Deldar

Isfahan University of Technology

1402-1403

SLR(1) Grammar

- **The SLR method begins with LR(0) items and LR(0) automata**
- ***Constructing an SLR-parsing table***

State i is constructed from I_i. The parsing actions for state i are determined as follows:

(a) If $[A \to \alpha \cdot a \beta]$ 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.

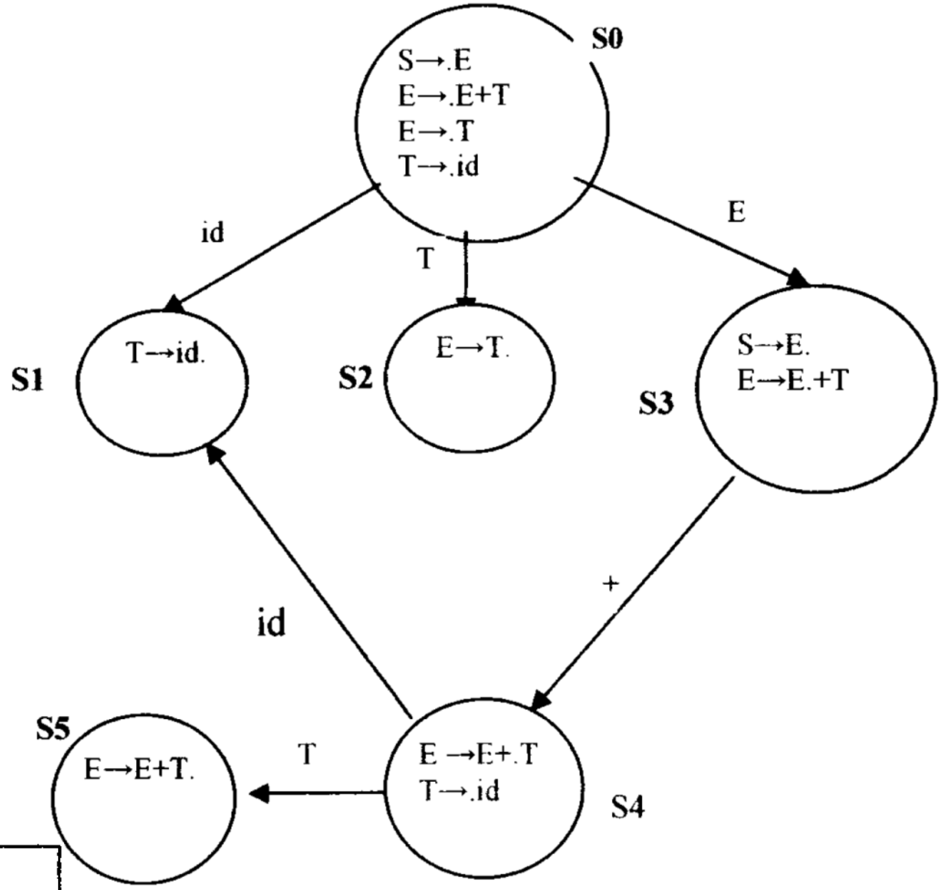
(b) If $[A \to \alpha \cdot]$ is in I_i, then set $\text{ACTION}[i, a]$ to "reduce $A \to \alpha$" for all a in $\text{FOLLOW}(A)$; here A may not be S'.

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

SLR(1) Grammar

- **Example**

1- S→E
2- E→E+T
3- E→T
4- T→id



حالات	action			goto	
	id	+	$	T	E
0	s1	error	error	2	3
1	error	r4	r4		
2	error	r3	r3		
3	error	s4	accept		
4	s1	error	error	5	
5	error	r2	r2		

SLR(1) Grammar

- **Example:** Parse string id+id

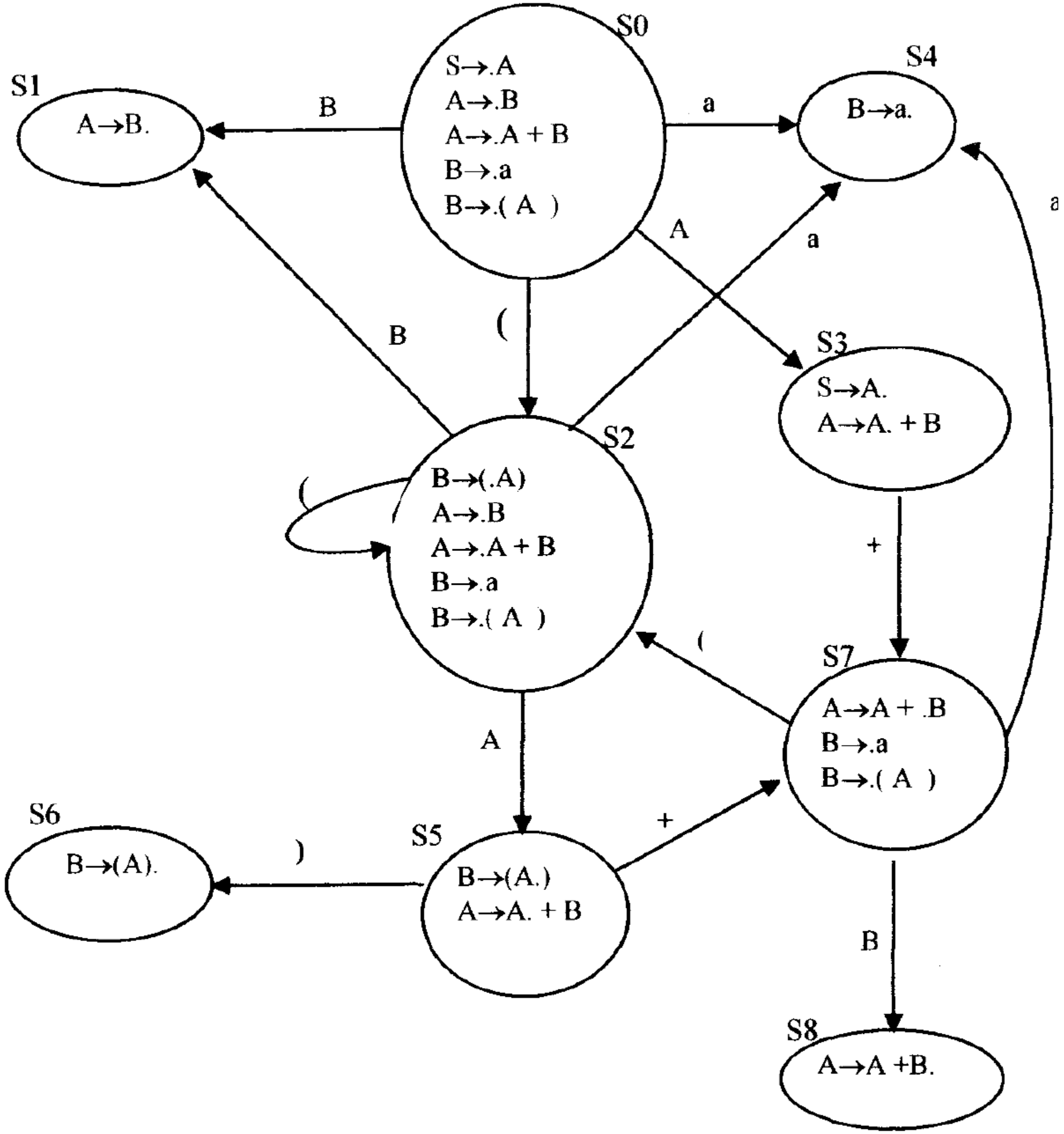
پشته	رشته ورودی	عملیات
0	id+id$	s1
0id1	+id$	r4: T→ id
0T	+id$	goto[0,T]=2
0T2	+id$	r3: E→ T
0E	+id$	goto[0,E]=3
0E3	+id$	s4
0E3+4	id$	s1
0E3+4id1	$	r4: T→ id
0E3+4T	$	goto[4,T]=5
0E3+4T5	$	r2: E→ E+T
0E	$	goto[0,E]=3
0E3	$	accept

SLR(1) Grammar

- **Example**

A→ B
A→ A + B
B→ a
B→ (A)

1- S→ A
2- A→ B
3- A→ A + B
4- B→ a
5- B→ (A)



SLR(1)

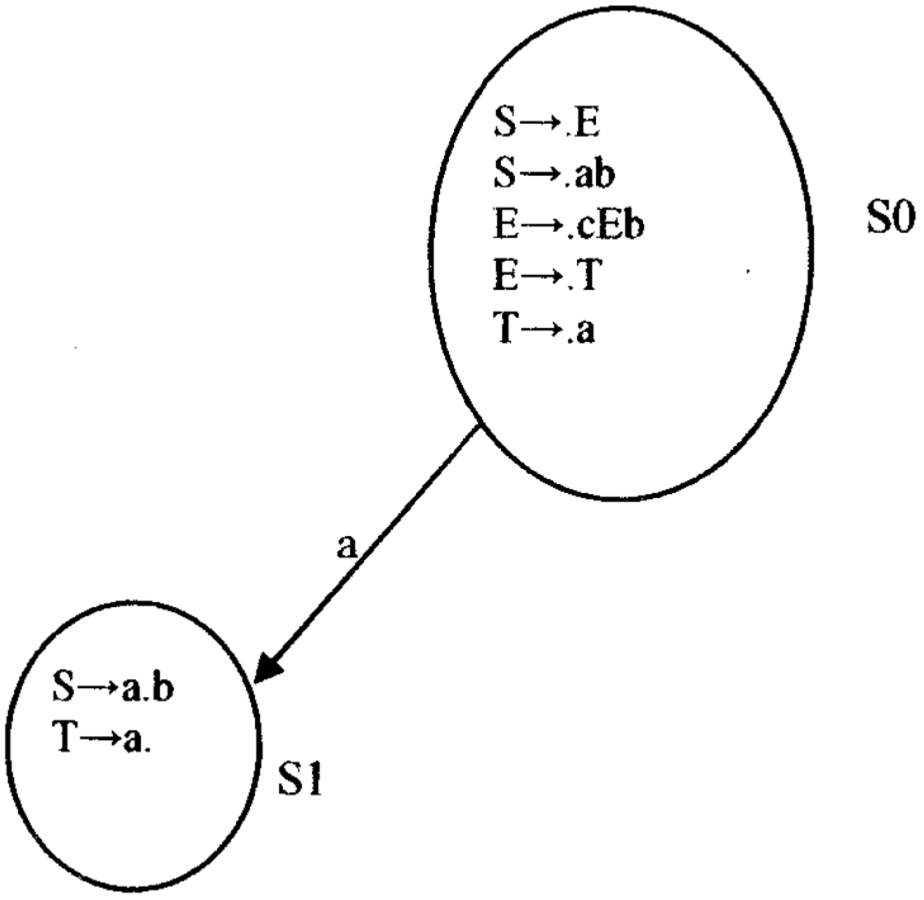
حالت	action					goto	
	a	+	(	)	$	A	B
0	s4		s2			3	1
1		r2		r2	r2		
2	s4		s2			5	1
3		s7			accept		
4		r4		r4	r4		
5		s7		s6			
6		r5		r5	r5		
7	s4		s2				8
8		r3		r3	r3		

پشته	رشته ورودی	اعمال انجام شده
0	(a+a)$	s2
0(2	a+a)$	s4
0(2a4	+a)$	r4: B→a
0(2B1	+a)$	r2: A→ B
0(2A5	+a)$	s7
0(2A5+7	a)$	s4
0(2A5+7a4	)$	r4: B→a
0(2A5+7B8	)$	r3: A→ A + B
0(2A5	)$	s6
0(2A5)6	$	r5: B→ (A)
0A3	$	accept

SLR(1) Grammar

- **Example: Shift/Reduce Conflict**
  - **The grammar is not SLR(1)**

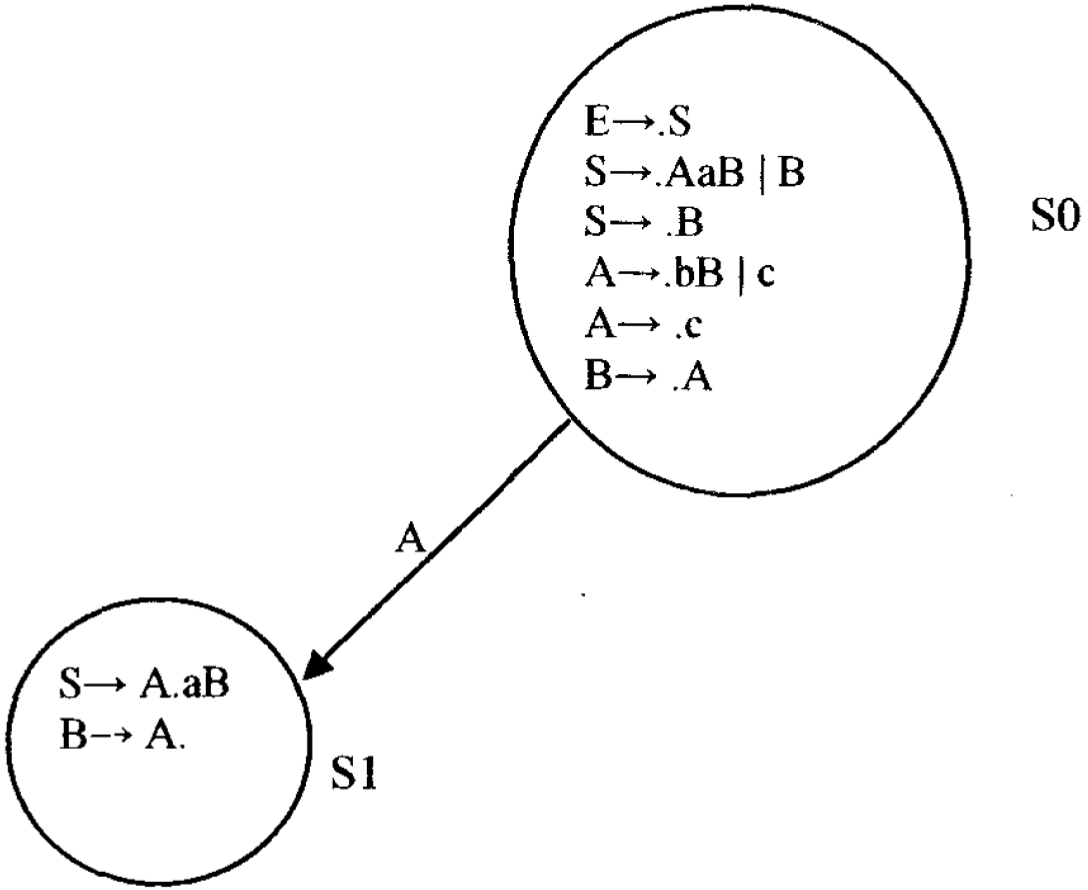
$S \rightarrow E \mid ab$
$E \rightarrow cEb \mid T$
$T \rightarrow a$



SLR(1) Grammar

- **Example: Shift/Reduce Conflict**
  - **The grammar is not SLR(1)**

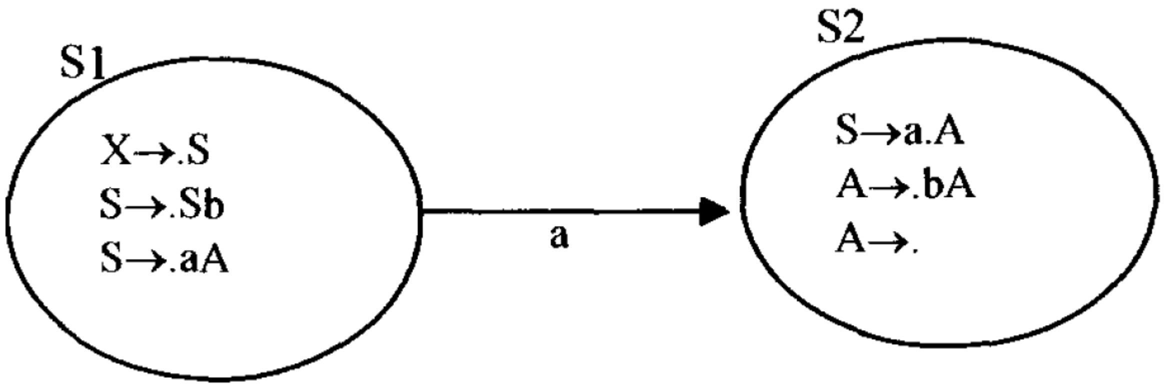
S→AaB | B
A→bB | c
B→A



SLR(1) Grammar

- **Example: Shift/Reduce Conflict**
  - **The grammar is not SLR(1)**

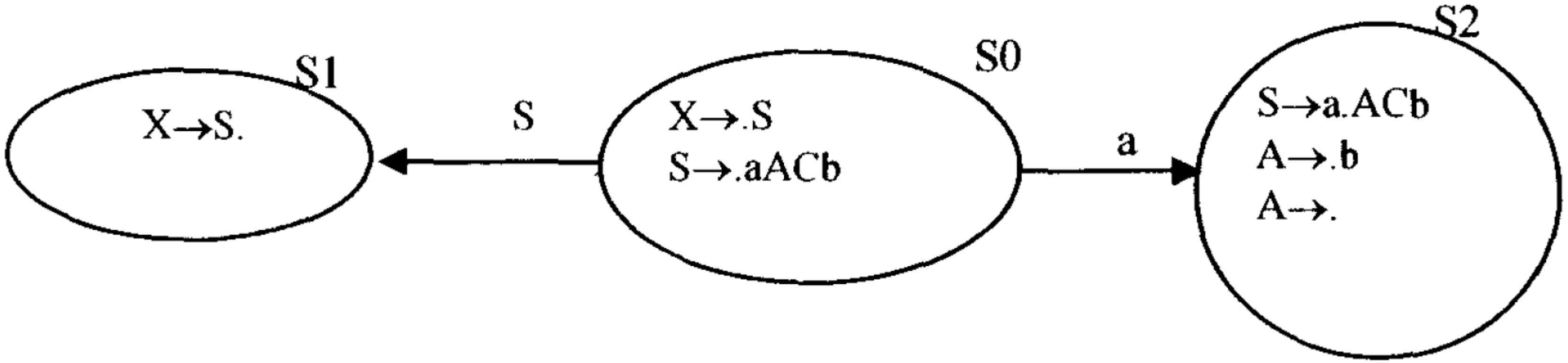
$S \rightarrow Sb \mid aA$
$A \rightarrow bA \mid \epsilon$



SLR(1) Grammar

- **Example: Shift/Reduce Conflict**
  - **The grammar is not SLR(1)**

$S \rightarrow aACb$
$A \rightarrow b \mid \epsilon$
$C \rightarrow cC \mid \epsilon$



SLR(1) Grammar

- ***Every SLR(1) grammar is unambiguous, but there are many unambiguous grammars that are not SLR(1)***

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

- ***Shift/Reduce conflict on input symbol =***

I_0: $S' \rightarrow \cdot S$
$S \rightarrow \cdot L = R$
$S \rightarrow \cdot R$
$L \rightarrow \cdot * R$
$L \rightarrow \cdot \mathbf{id}$
$R \rightarrow \cdot L$

I_1: $S' \rightarrow S \cdot$

I_2: $S \rightarrow L \cdot = R$
$R \rightarrow L \cdot$

I_3: $S \rightarrow R \cdot$

I_4: $L \rightarrow * \cdot R$
$R \rightarrow \cdot L$
$L \rightarrow \cdot * R$
$L \rightarrow \cdot \mathbf{id}$

I_5: $L \rightarrow \mathbf{id} \cdot$

I_6: $S \rightarrow L = \cdot R$
$R \rightarrow \cdot L$
$L \rightarrow \cdot * R$
$L \rightarrow \cdot \mathbf{id}$

I_7: $L \rightarrow *R \cdot$

I_8: $R \rightarrow L \cdot$

I_9: $S \rightarrow L = R \cdot$

Constructing LR(1) Sets of Items

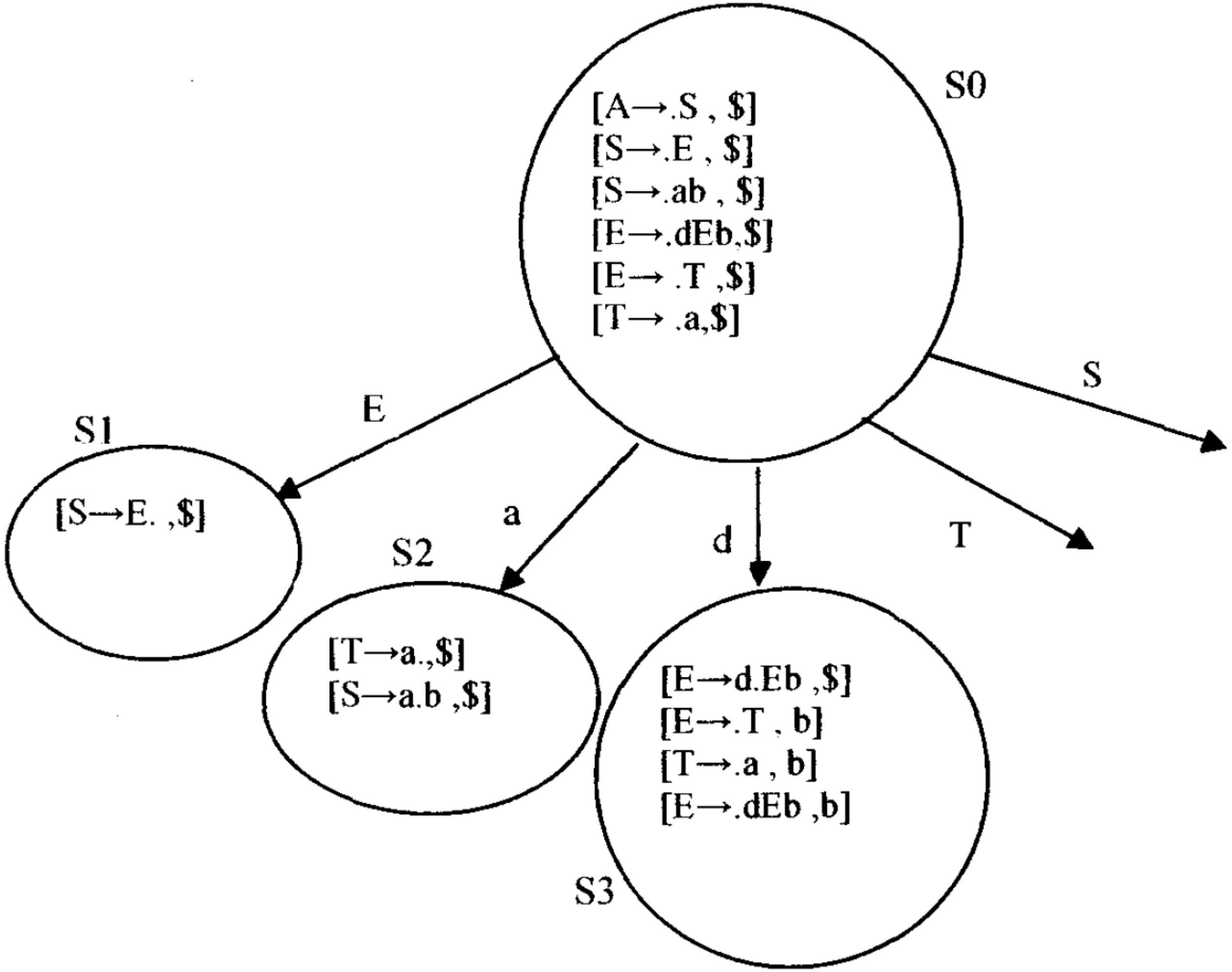
```
SetOfItems CLOSURE(I) {
    repeat
        for ( each item [A → α·Bβ, a] in I )
            for ( each production B → γ in G' )
                for ( each terminal b in FIRST(βa) )
                    add [B → ·γ, b] to set I;
    until no more items are added to I;
    return I;
}

SetOfItems GOTO(I, X) {
    initialize J to be the empty set;
    for ( each item [A → α·Xβ, a] in I )
        add item [A → αX·β, a] to set J;
    return CLOSURE(J);
}
```

LR(1) Grammar

S→ E | ab
E→ dEb | T
T→ a

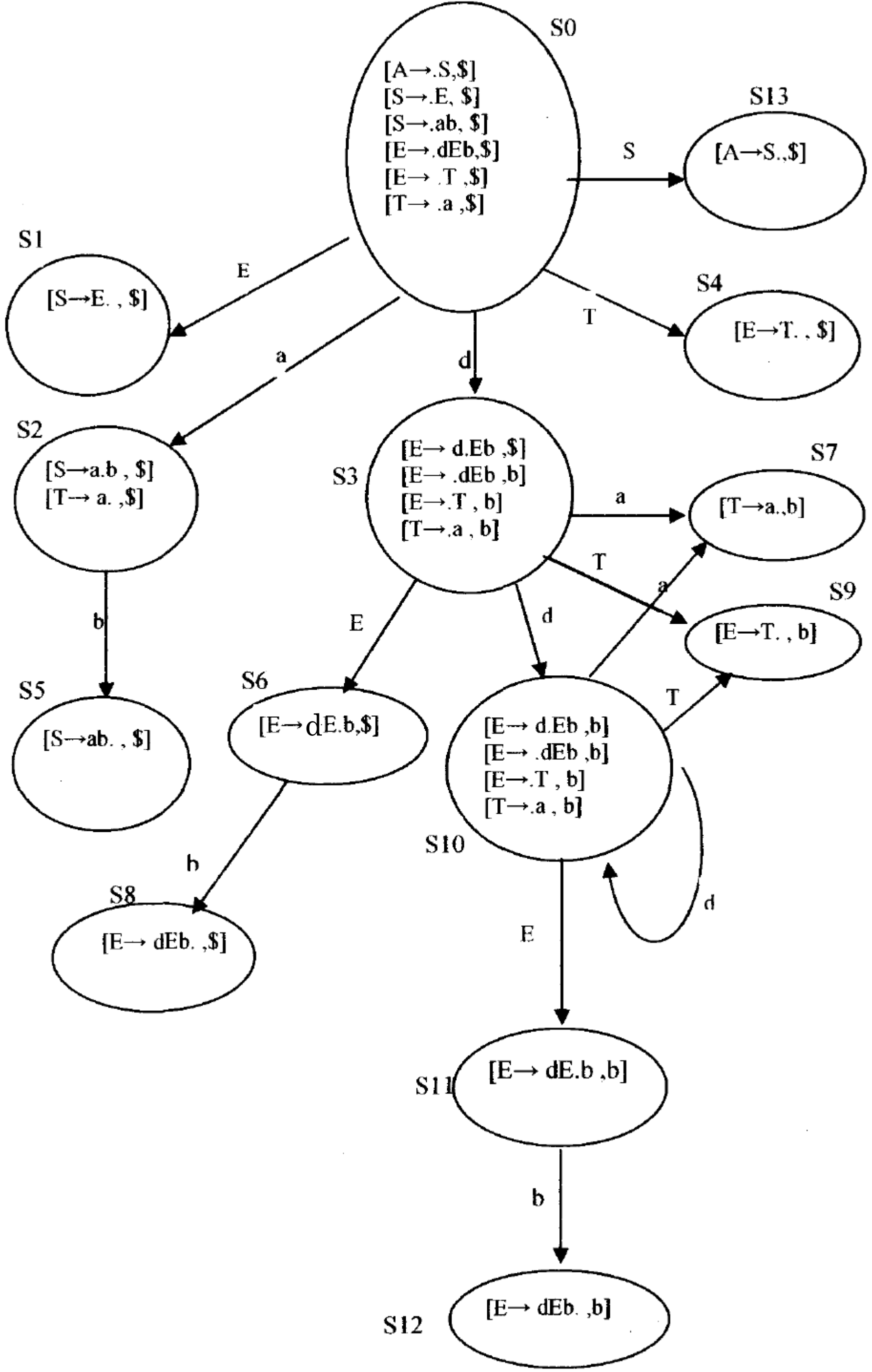
1- A→S
2- S→ E
3- S→ ab
4- E→ dEb
5- E→ T
6- T→ a



LR(1) Grammar

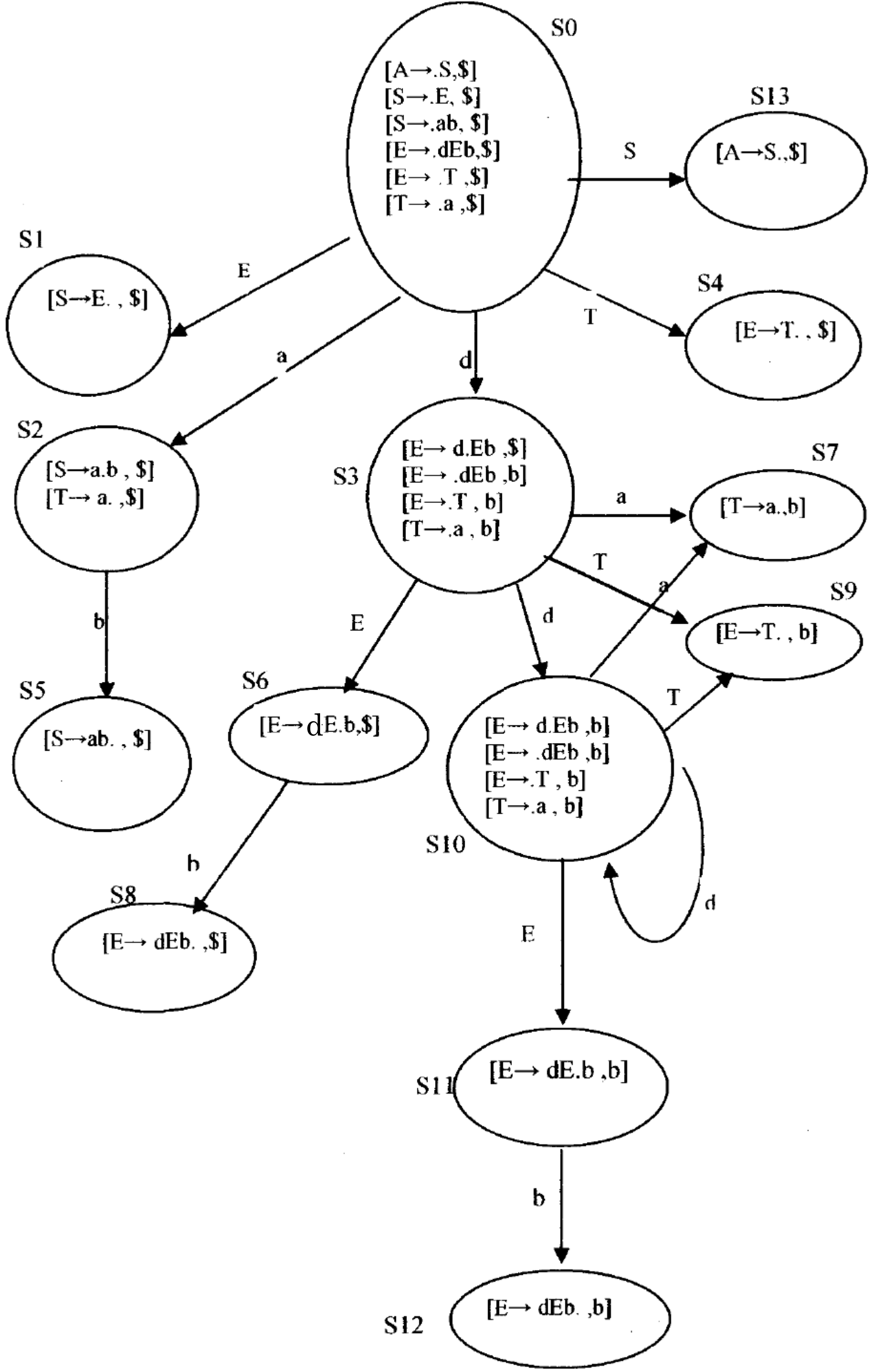
S→ E | ab
E→ dEb | T
T→ a

1- A→S
2- S→ E
3- S→ ab
4- E→ dEb
5- E→ T
6- T→ a



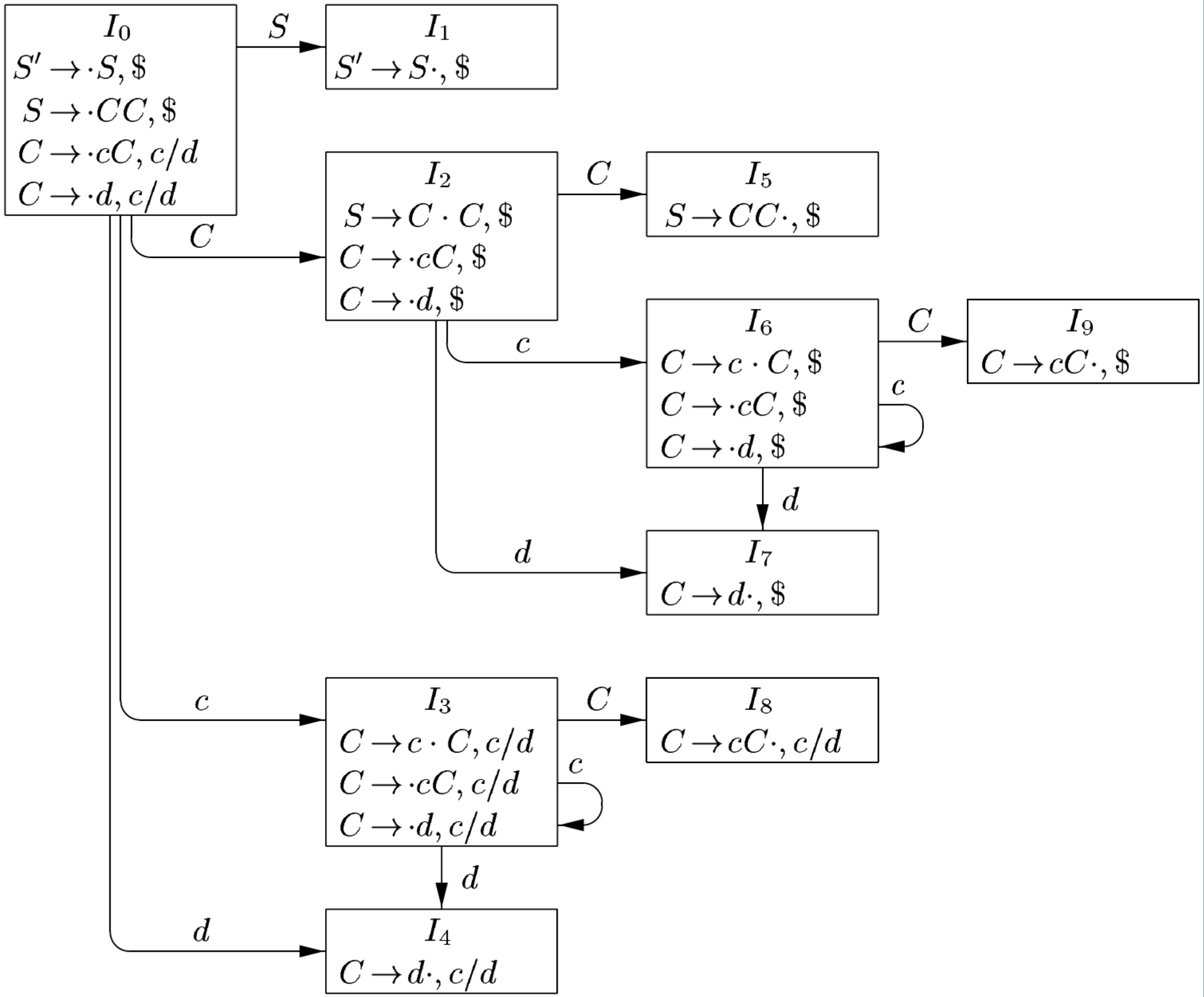
LR(1) Grammar

حالات	action				goto		
	a	b	d	$	E	T	S
0	s2		s3		1	4	13
1				r2			
2		s5		r6			
3	s7		s10		6	9	
4				r5			
5				r3			
6		s8					
7		r6					
8				r4			
9		r5					
10	s7		s10		11	9	
11		s12					
12		r4					
13				accept			



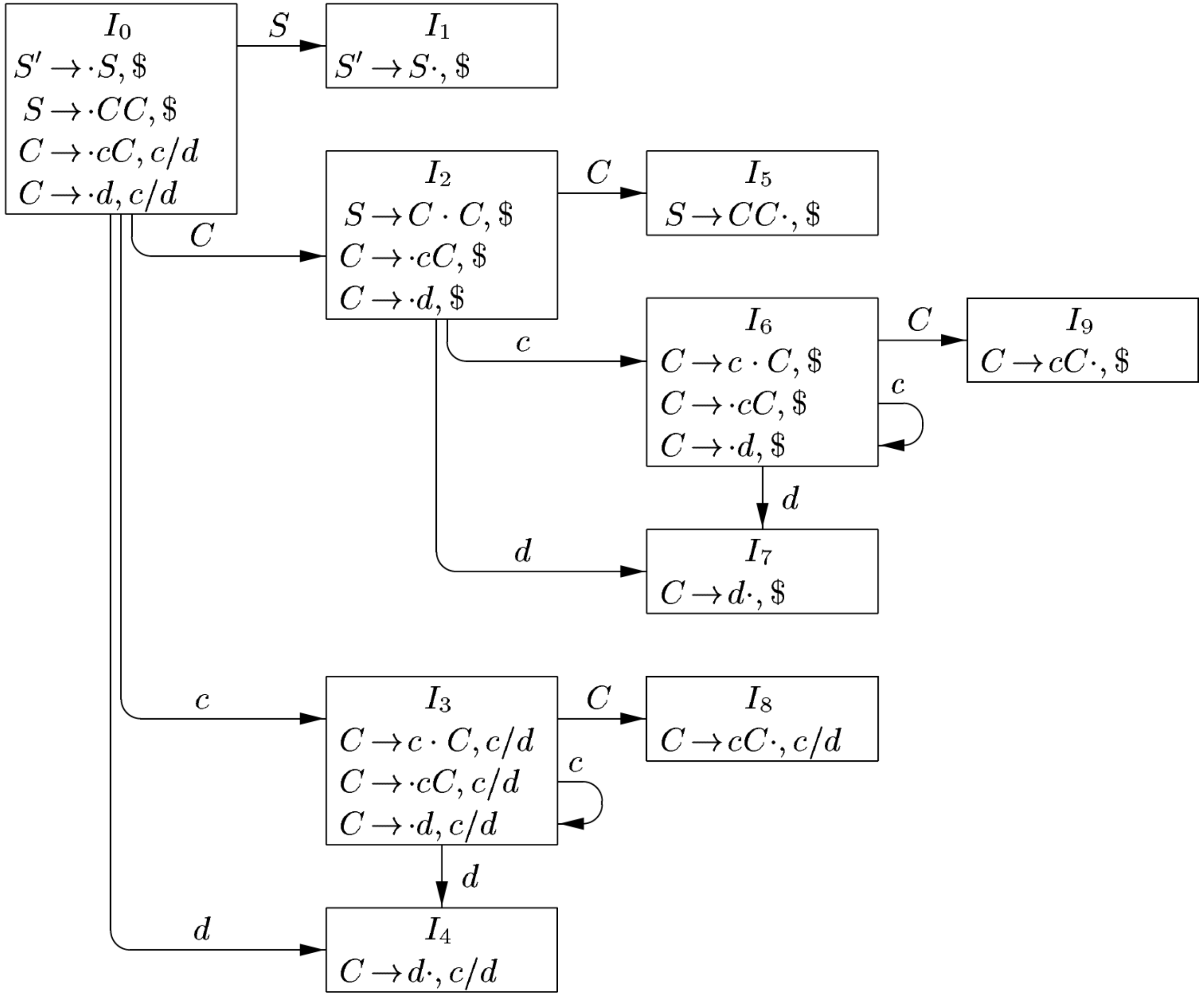
LR(1) Grammar

$$
\begin{array}{lcl}
S' & \rightarrow & S \\
S & \rightarrow & C\ C \\
C & \rightarrow & c\ C \ \mid\ d
\end{array}
$$



LR(1) Grammar

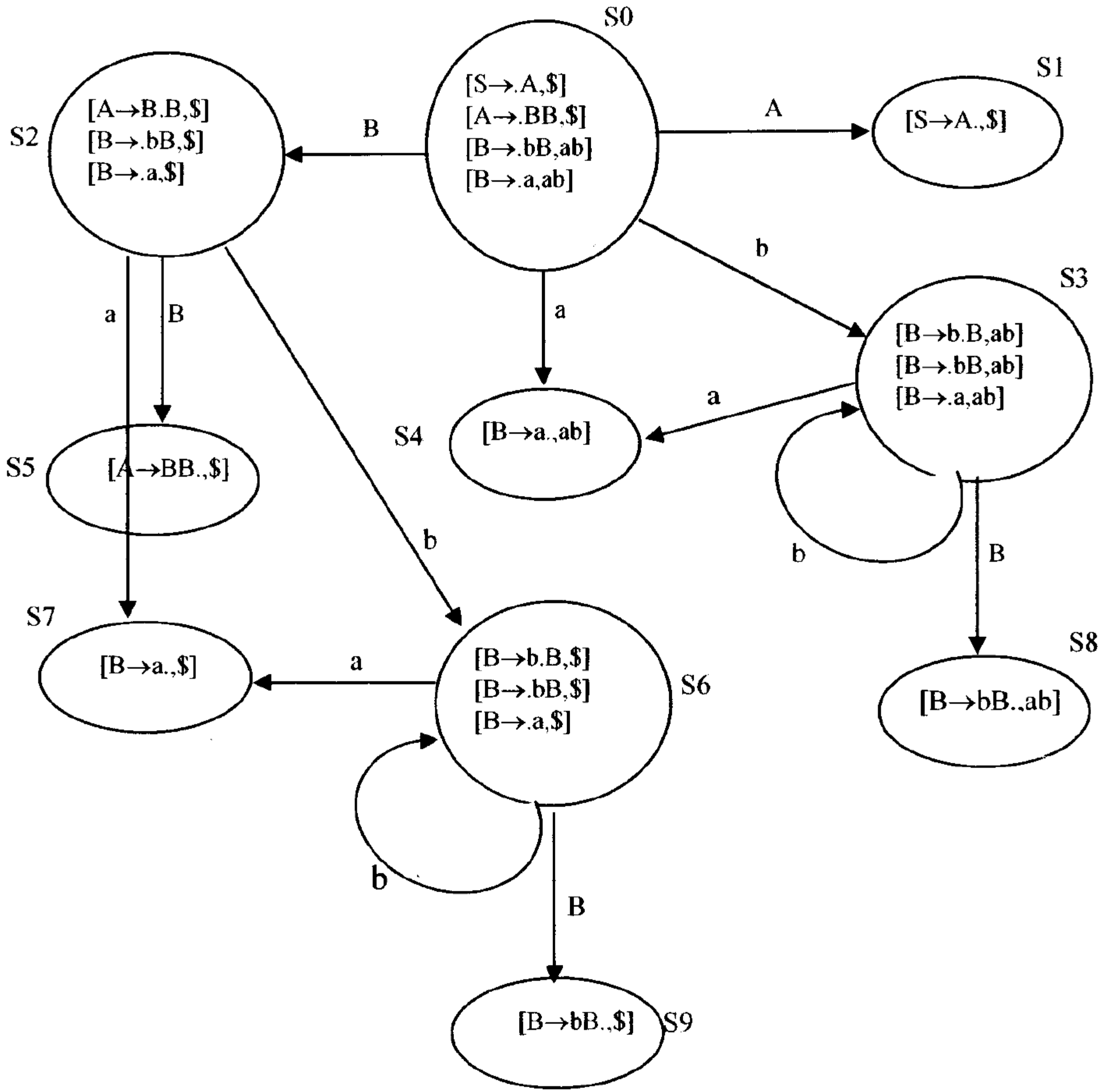
STATE	ACTION			GOTO	
	c	d	\$	S	C
0	s3	s4		1	2
1			acc		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		



LR(1) Grammar

A→BB
B→bB|a

0- S→ A
1- A→ BB
2- B→ bB
3- B→ a



LR(1) Grammar

حالت	action			goto	
	b	a	$	A	B
0	s3	s4		1	2
1			accept		
2	s6	s7			5
3	s3	s4			8
4	r3	r3			
5			r1		
6	s6	s7			9
7			r3		
8	r2	r2			
9			r2		

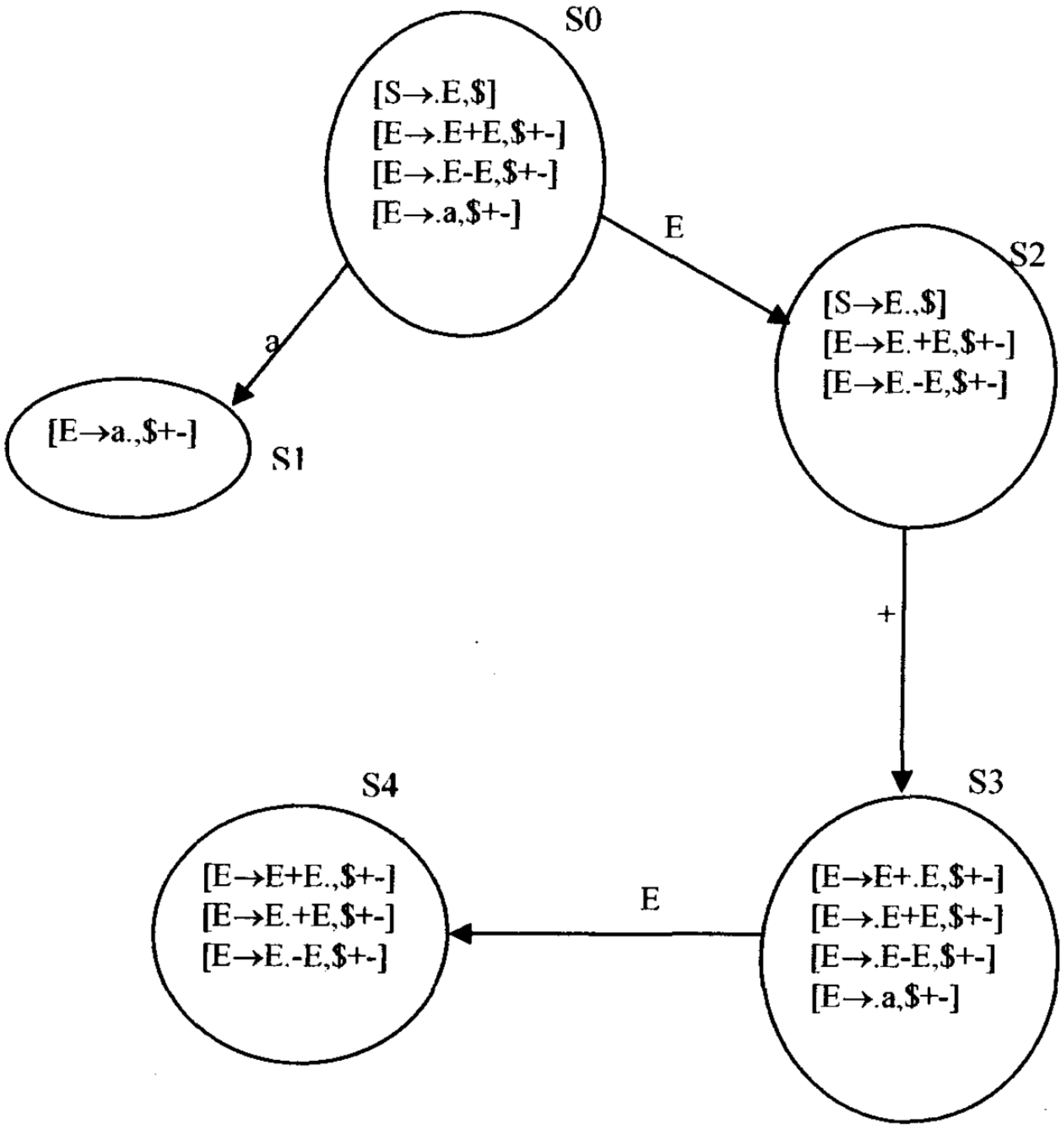
LR(1) Grammar

- **Example: Shift/Reduce Conflict**
  - The grammar is not LR(1)

E→E+E|E-E|a

↓

1- S→E
2- E→E+E
3- E→E-E
4- E→a



Constructing LALR Parsing Tables

- **LALR (LookAhead LR)**
- **This method is often used in practice, because:**
  - **The tables obtained by it are considerably smaller than the canonical LR tables**
  - **Most common syntactic constructs of programming languages can be expressed conveniently by an LALR grammar**
- **The SLR and LALR tables for a grammar always have the same number of states**
- **Example: For a language like C:**
  - The SLR and LALR tables have typically ***several hundred states***
  - The canonical LR table would typically have ***several thousand states***

Constructing LALR Parsing Tables

- **We look for sets of LR(1) items having the same core, and merge these sets with common cores into one set of items**

- *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***

- **But it is possible that a merger will produce a** ***reduce/reduce*** **conflict**

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

2. For each core present among the set of LR(1) items, find all sets having that core, and replace these sets by their union.

LALR(1) Grammar

- **Example**

$$\begin{aligned} I_{36}: \quad & C \to c{\cdot}C,\ c/d/\$ \\ & C \to {\cdot}cC,\ c/d/\$ \\ & C \to {\cdot}d,\ c/d/\$ \end{aligned}$$

$$I_{47}: \quad C \to d{\cdot},\ c/d/\$$$

$$I_{89}: \quad C \to cC{\cdot},\ c/d/\$$$

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		

STATE	ACTION			GOTO	
	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		

LALR(1) Grammar

- **Example: Reduce/Reduce Conflict**
  - **The grammar is not LALR(1)**

$$
\begin{array}{lcl}
S' & \rightarrow & S \\
S & \rightarrow & a\ A\ d \mid b\ B\ d \mid a\ B\ e \mid b\ A\ e \\
A & \rightarrow & c \\
B & \rightarrow & c
\end{array}
$$

$$\{[A \rightarrow c\cdot,\ d], [B \rightarrow c\cdot,\ e]\}$$
$$\{[A \rightarrow c\cdot, e], [B \rightarrow c\cdot,\ d]\}$$

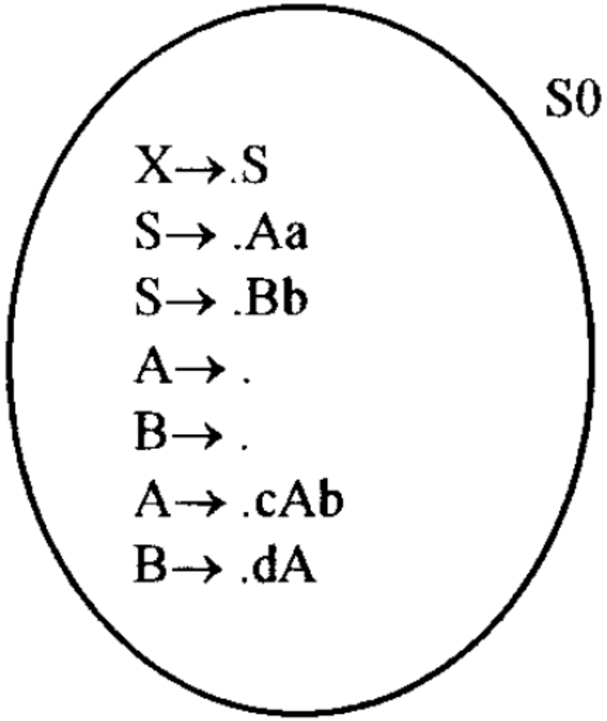
$$\Downarrow$$

$$
\begin{array}{l}
A \rightarrow c\cdot,\ d/e \\
B \rightarrow c\cdot,\ d/e
\end{array}
$$

LALR(1) Grammar

- **Example:**
  - **Reduce/Reduce conflict in SLR(1) automata**
    - Not SLR(1)

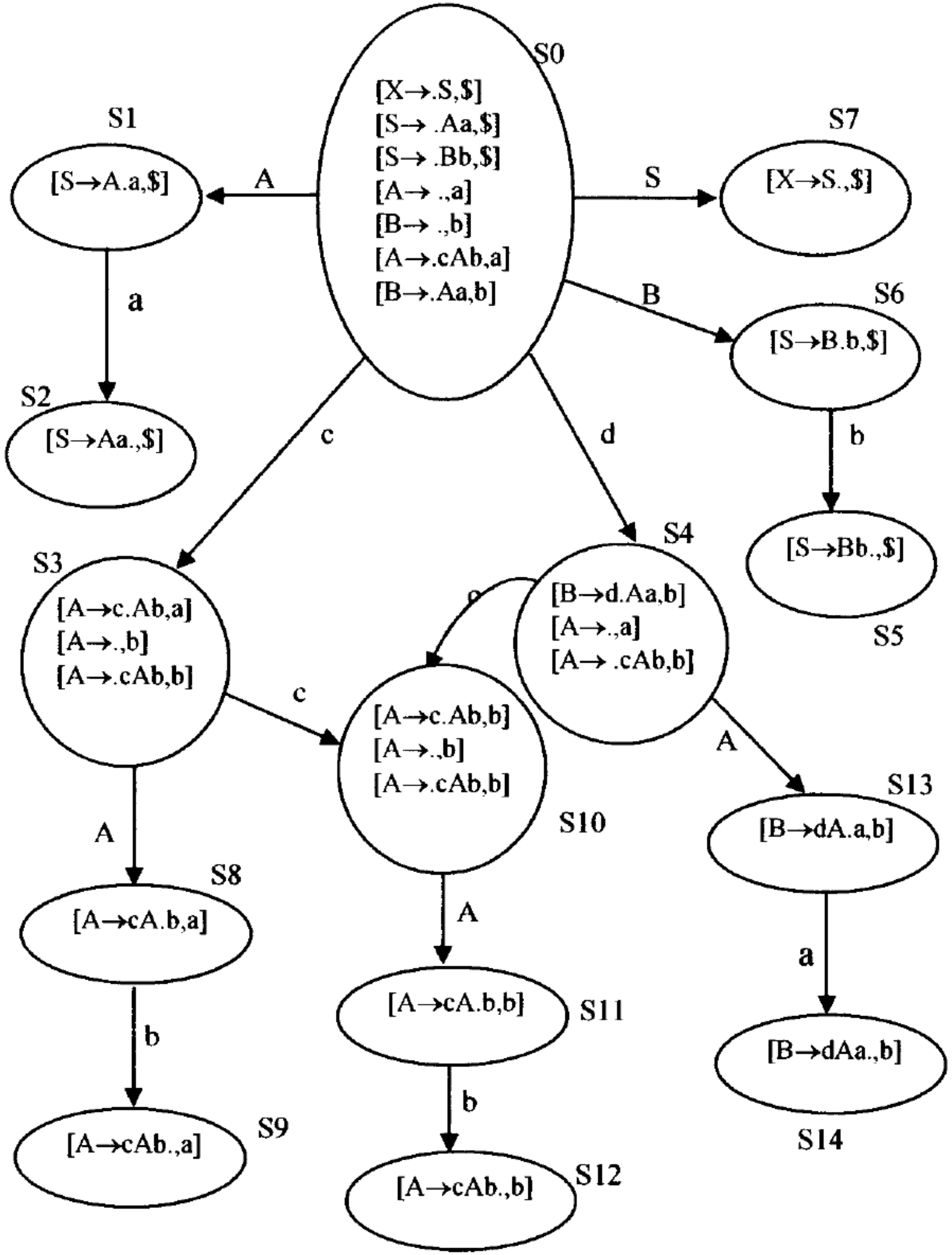
$S \to Aa$
$S \to Bb$
$A \to \epsilon$
$B \to \epsilon$
$A \to cAb$
$B \to dAa$



LALR(1) Grammar

- **Example:**
    - No conflict in LR(1) automata
        - LR(1)

S→ Aa
S→ Bb
A→ ∈
B→ ∈
A→ cAb
B→ dAa



LALR(1) Grammar

- **Example:**
  - No conflict in LALR(1) automata
    - LALR(1)

S8:[A→cA.b,a]	S11:[A→cA.b,b]	S8,11:[A→cA.b,ab]
S9:[A→cAb.,a]	S12:[A→cAb.,b]	S9,12:[A→cAb.,ab]
S3:[A→c.Ab,a] [A→.,b] [A→.cAb,b]	S10:[A→c.Ab,b] [A→.,b] [A→.cAb,b]	S3,10:[A→c.Ab,ab] [A→.,b] [A→.cAb,b]