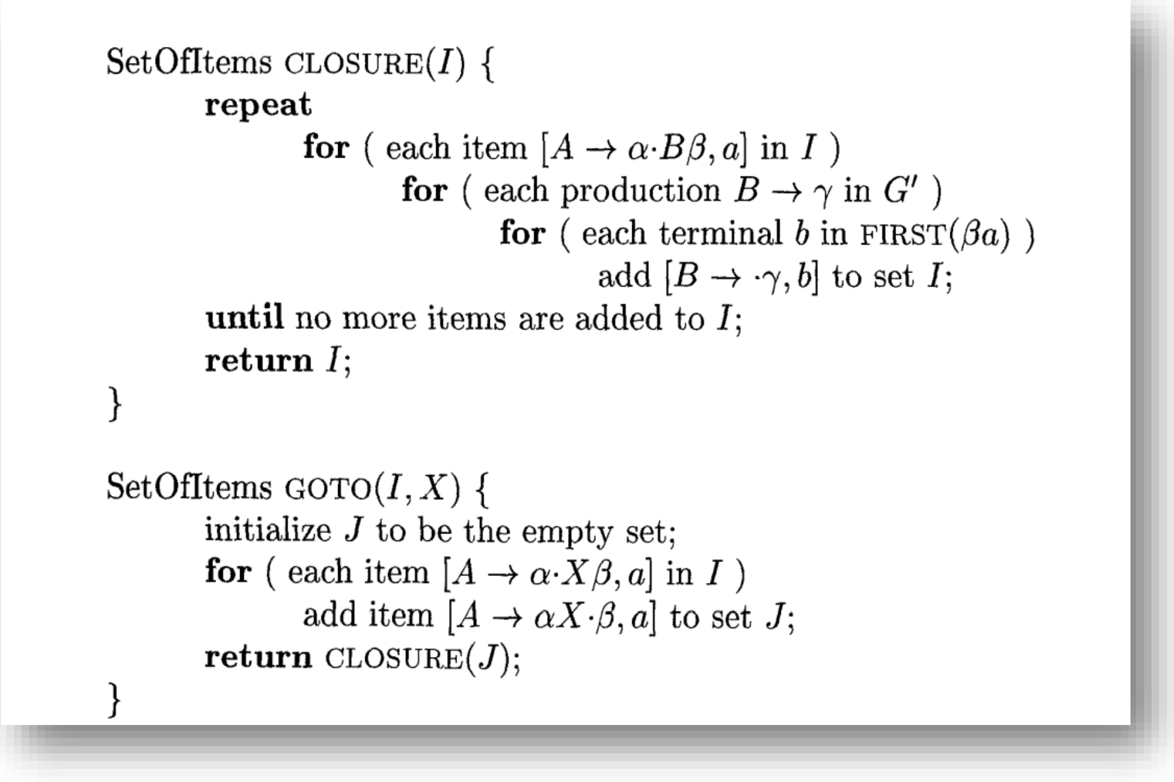
# Syntax Analysis

## LALR(1) Parsers

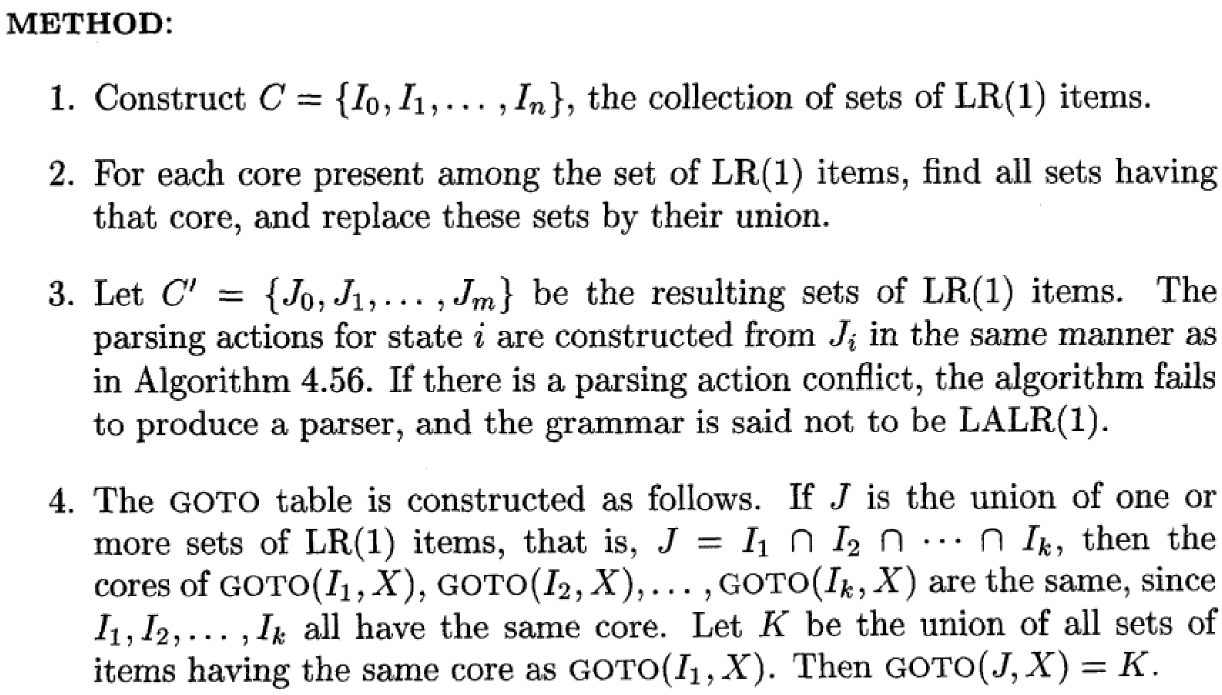
### Constructing Canonical Collections of LR(1) items



```
SetOfItems CLOSURE($I$) {
    repeat
        for ( each item $[A \to \alpha \cdot B\beta, a]$ in $I$ )
            for ( each production $B \to \gamma$ in $G'$ )
                for ( each terminal $b$ in FIRST($\beta a$) )
                    add $[B \to \cdot\gamma, b]$ 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 $[A \to \alpha \cdot X\beta, a]$ in $I$ )
        add item $[A \to \alpha X \cdot \beta, a]$ to set $J$;
    return CLOSURE($J$);
}
```

### Construction of LALR parsing table:



**METHOD:**

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.

3. Let $C' = \{J_0, J_1, \ldots, J_m\}$ be the resulting sets of LR(1) items. The parsing actions for state $i$ are constructed from $J_i$ in the same manner as in Algorithm 4.56. If there is a parsing action conflict, the algorithm fails to produce a parser, and the grammar is said not to be LALR(1).

4. The GOTO table is constructed as follows. If $J$ is the union of one or more sets of LR(1) items, that is, $J = I_1 \cap I_2 \cap \cdots \cap I_k$, then the cores of GOTO($I_1, X$), GOTO($I_2, X$), $\ldots$ , GOTO($I_k, X$) are the same, since $I_1, I_2, \ldots, I_k$ all have the same core. Let $K$ be the union of all sets of items having the same core as GOTO($I_1, X$). Then GOTO($J, X$) $= K$.

Ex: Consider the given Grammar:

**S→CC**

**C→cC|d**

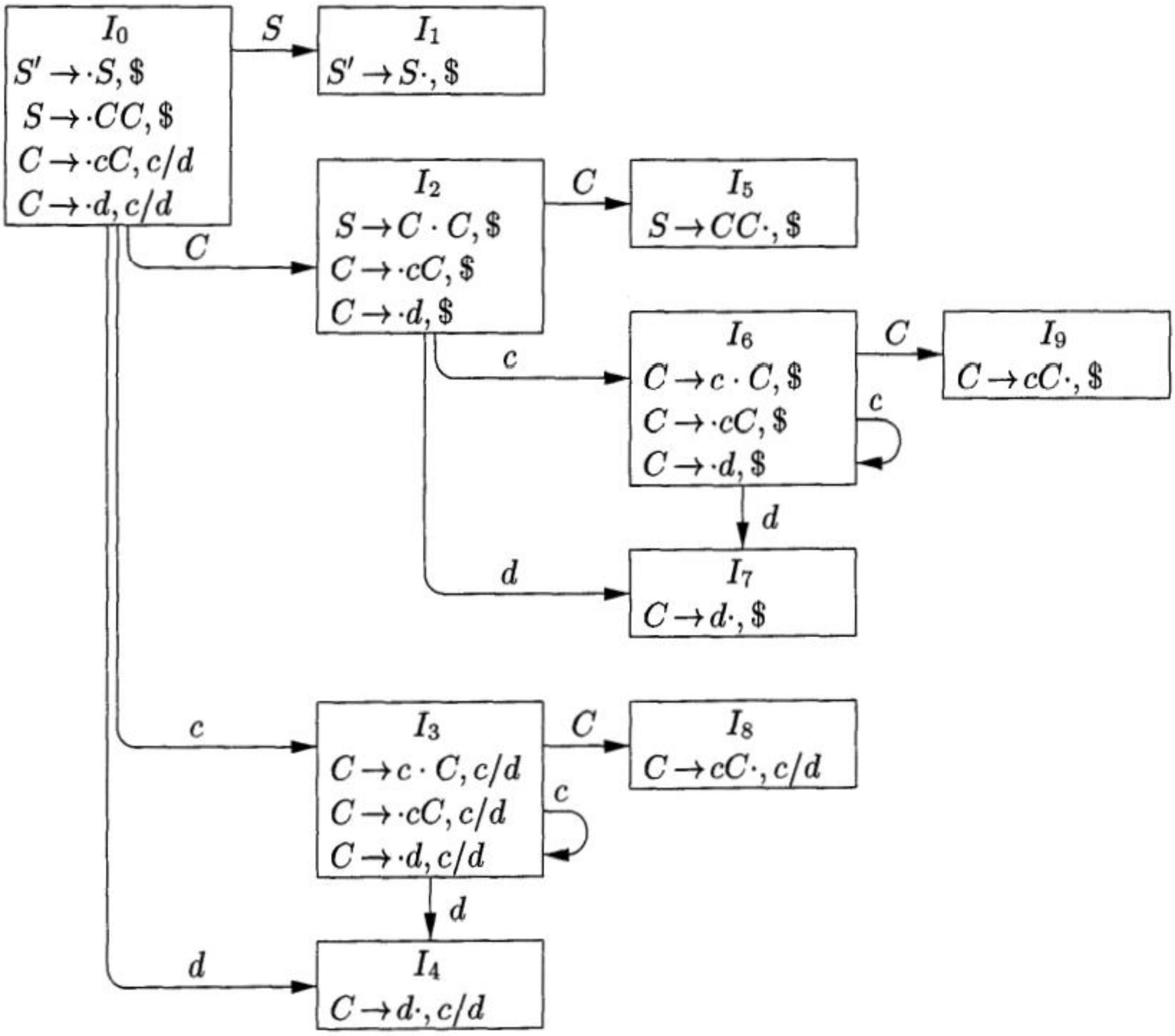
**Step1:** Augmented grammar:

**S'→S**

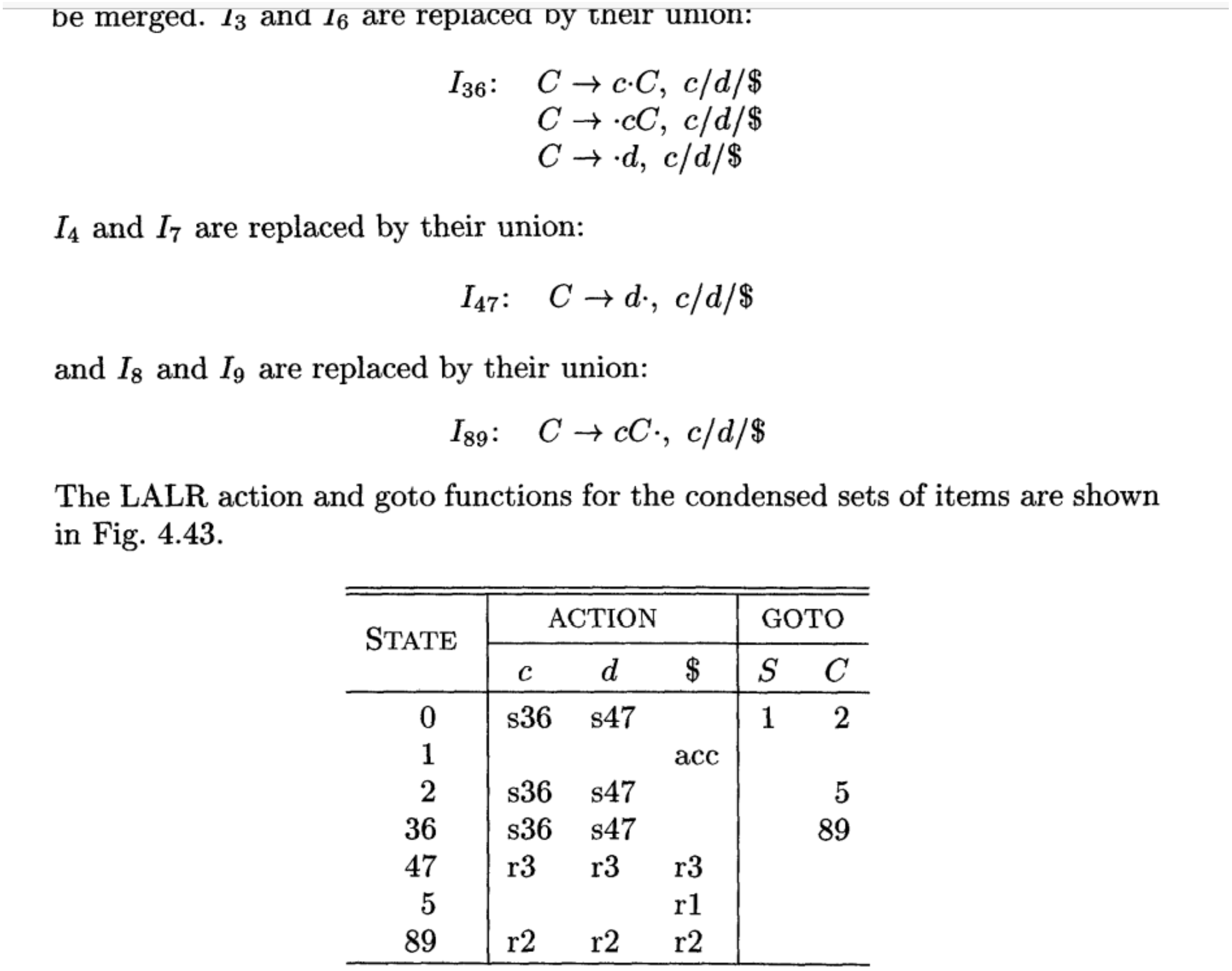
**S→CC**

**C→cC|d**

**Step 2: Finding the canonical set of LR(1) items:**



**Step 3: Parsing table:**



be merged. $I_3$ and $I_6$ are replaced by their union:

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

$I_4$ and $I_7$ are replaced by their union:

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

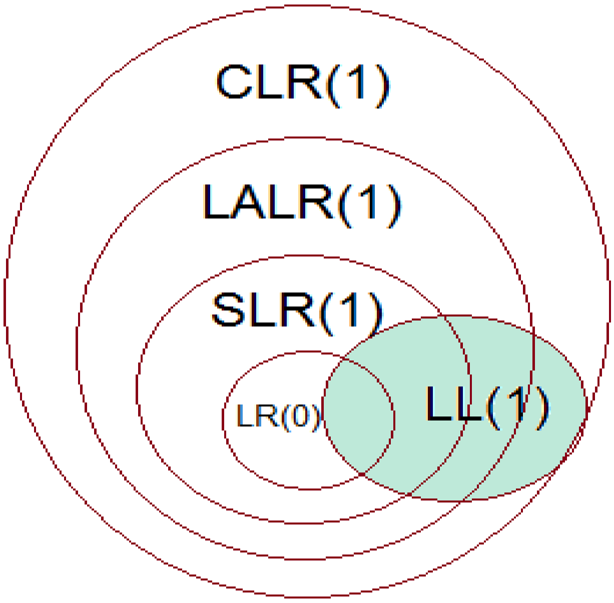
and $I_8$ and $I_9$ are replaced by their union:

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

The LALR action and goto functions for the condensed sets of items are shown in Fig. 4.43.

| 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 | | |

## Relationship among all the parsers:



# Conflicts in Shift-Reduce Parsers

**1. Shift-Reduce:** A *shift-reduce* error occurs when the parser cannot decide whether to continue shifting or to reduce (using a different production rule).

**2. Reduce-Reduce:** A *Reduce-Reduce* error occurs when the parser has to choose between more than one equally acceptable productions.

## **Conflicts in different items:

**1. LR(0)-** In the same state if we have shift and reduce moves combined, then SR Conflict.

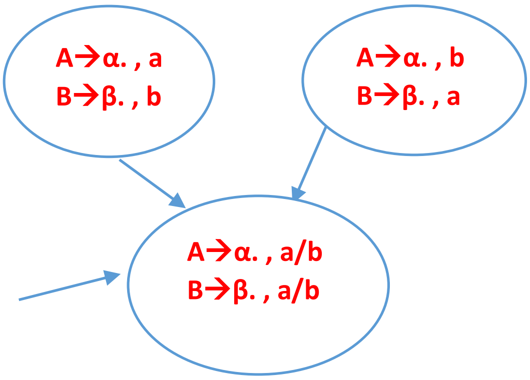
i.e., $A \rightarrow .\alpha$
$B \rightarrow \beta.$

**2. LR(1)-**

**SR Conflicts**: $A \rightarrow \alpha.a\ \beta\ ,\ c/d$ **[Shift on a]**
$B \rightarrow \gamma.\ ,\ a/\$$ **[Reduce by a]**

**RR Conflicts:** $A \rightarrow \alpha.\ ,\ a$ **[Reduce by a]**
$B \rightarrow \alpha.\ ,\ a$ **[Reduce by a]**

- If there is either SR or RR conflict then the grammar is not CLR(1) and also not LALR(1).
- There is a chance a grammar to be CLR (1) but not LALR (1), if in LR (1) items there may not have any conflict but after merging the states there may be a conflict on look ahead.



**Both of these productions will be placed on the table in the a and b columns. So not in LALR(1).**

- If there is not any SR conflicts in CLR(1) then also not in LALR(1).
- If there is not any RR conflicts in CLR(1) but there could be RR conflict in LALR(1).

## Identify a grammar is in:

**LL(1):** Grammar is unambiguous.
**LR(0):** In LR(0) items there is no SR or RR conflict.
**SLR(1):** In LR(0) there is no SR or RR conflict.
**CLR(1):** In LR(1) items there is no SR or RR conflict for look ahead symbol.
**LALR(1):** In CLR(1) there is no SR or RR conflict and also no merge conflict.

Exercises:

1. Check the following grammars are in LL(1),LR(0),SLR(1),CLR(1),LALR(1) or not.

i) S→AaAb | BbBa

A→ ϵ

B→ ϵ

ii) S→Aa | bAc | dc | bda

A→d

iii) S→Aa | bAc | Bc | bBa

A→d

B→d

2. Find the number of SR and RR conflicts in dfa with LR(0) items.

S→SS | a | ϵ

3. S→(S) | a , where no. of states in SLR(1)=n1 , LR(1)=n2 and LALR(1)=n3.

a) n1<n2<n3

b) n1=n3<n2

c) n1=n2=n3

d) n1>=n3>=n2