Syntax Analysis

LR Parsers

CLR(1) and LALR(1)

LR(1) Items

An ***LR(1) item*** is defined as a production rule which has a dot(**.**) on its RHS with look ahead symbols. The dot represents the input symbols that have been read and input symbols waiting to be read.

Constructing Canonical Collections of LR(1) 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);
}
```

Algorithm 4.56: Construction of canonical-LR parsing tables.

INPUT: An augmented grammar G'.

OUTPUT: The canonical-LR parsing table functions ACTION and GOTO for G'.

METHOD:

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 GOTO$(I_i, a) = I_j$, then set 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 ACTION$[i, a]$ to "reduce $A \to \alpha$."
   (c) If $[S' \to S \cdot, \$]$ is in I_i, then set 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 GOTO$(I_i, A) = I_j$, then 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 $[S' \to \cdot S, \$]$.

Ex: Consider the given Grammar:

S→CC

C→cC|d

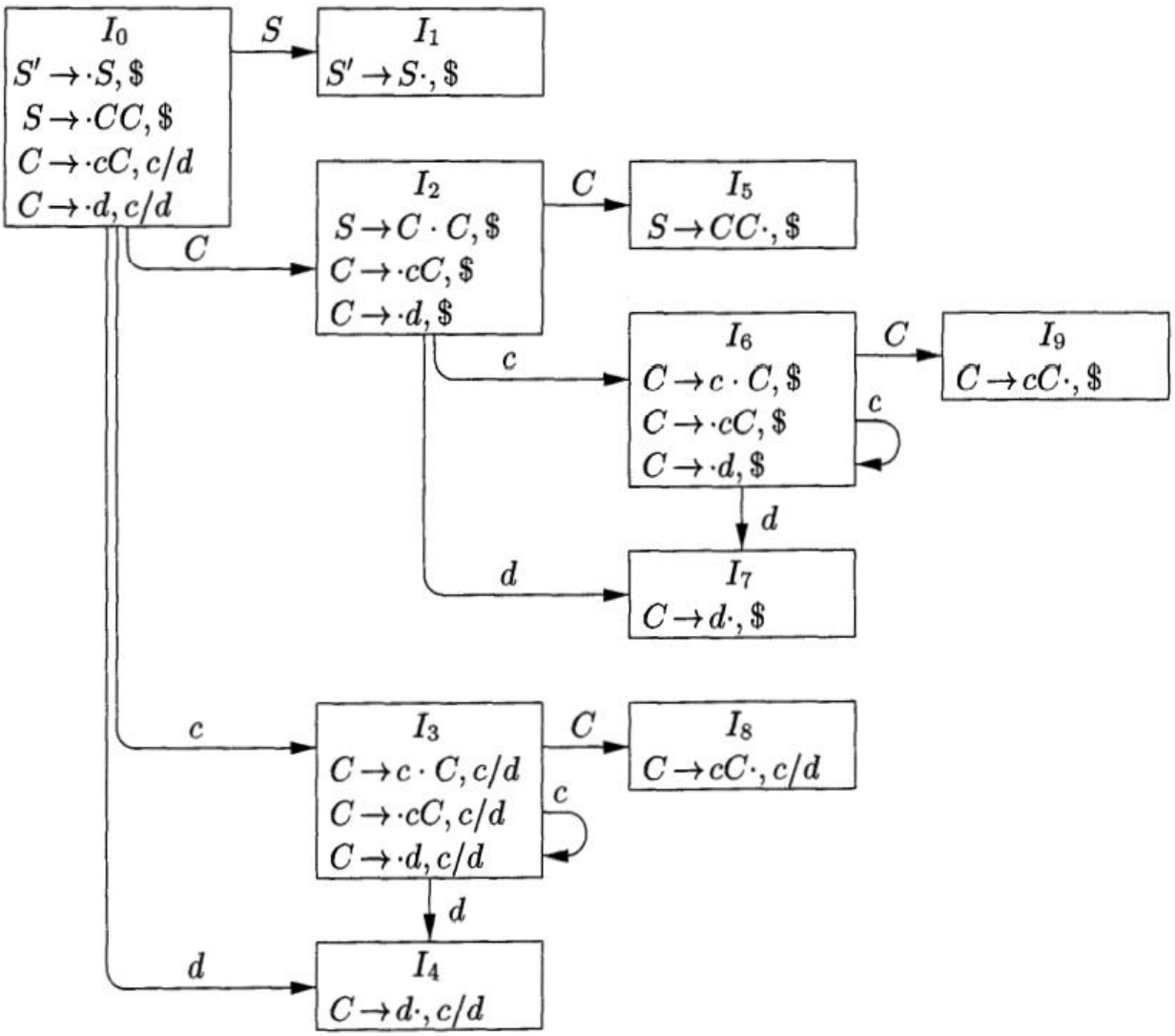
Step1: Augmented grammar:

S'→S

S→CC

C→cC|d

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



Step 3: Construction of CLR 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		

Do the following:

1. G→ S
   S→ E=E
   S→ f
   E→ T
   E→ E+T
   T→ f
   T→ T*f

2. S→AaAb | BbBa
   A→ϵ
   B→ϵ

3. S→ L = R|R
   L→*R| id
   R→L

4. S→ aAd | bBd | aBe | bAe
   A→ c
   B→c

5. S→ Aa | bAc | Bc | bBa
   A→d
   B→d

**In CLR parser if two states differ only in look ahead then we combine those states in LALR parser.