



4. Calculate **GOTO**( $I, X$ ) where  $I = \{S \rightarrow \bullet SN, SN \rightarrow SN \bullet baa\}$  and  $X = baa$ .

5. Construct the **Canonical LR(0) Collection** using the following algorithm:

```
void items( $G'$ ) {
     $C = \text{CLOSURE}(\{[S' \rightarrow \cdot S]\});$ 
    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  $C$  on a round;
}
```

6. Draw the resulting **LR(0) Automaton**.

7. Construct the **SLR-Parsing Table** using the following algorithm

1. Construct  $C = \{I_0, I_1, \dots, I_n\}$ , the collection of sets of LR(0) items for  $G'$ .
2. State  $i$  is constructed from  $I_i$ . The parsing actions for state  $i$  are determined as follows:
  - (a) If  $[A \rightarrow \alpha \cdot a \beta]$  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  $[A \rightarrow \alpha \cdot]$  is in  $I_i$ , then set  $\text{ACTION}[i, a]$  to "reduce  $A \rightarrow \alpha$ " for all  $a$  in  $\text{FOLLOW}(A)$ ; here  $A$  may not be  $S'$ .
  - (c) If  $[S' \rightarrow S \cdot]$  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 SLR(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  $[S' \rightarrow \cdot S]$ .

STATE	ACTION		GOTO	
	baa	\$	S	SN
0				
1				
2				
3				

8. Is this grammar **SLR**?

9. Show me the actions of the resulting parser for the **input string** "baa baa baa baa" (bbbb)

	Stack	Input	Action	Output
1				
2				
3				
4				
5				
6				
7				
8				
9				