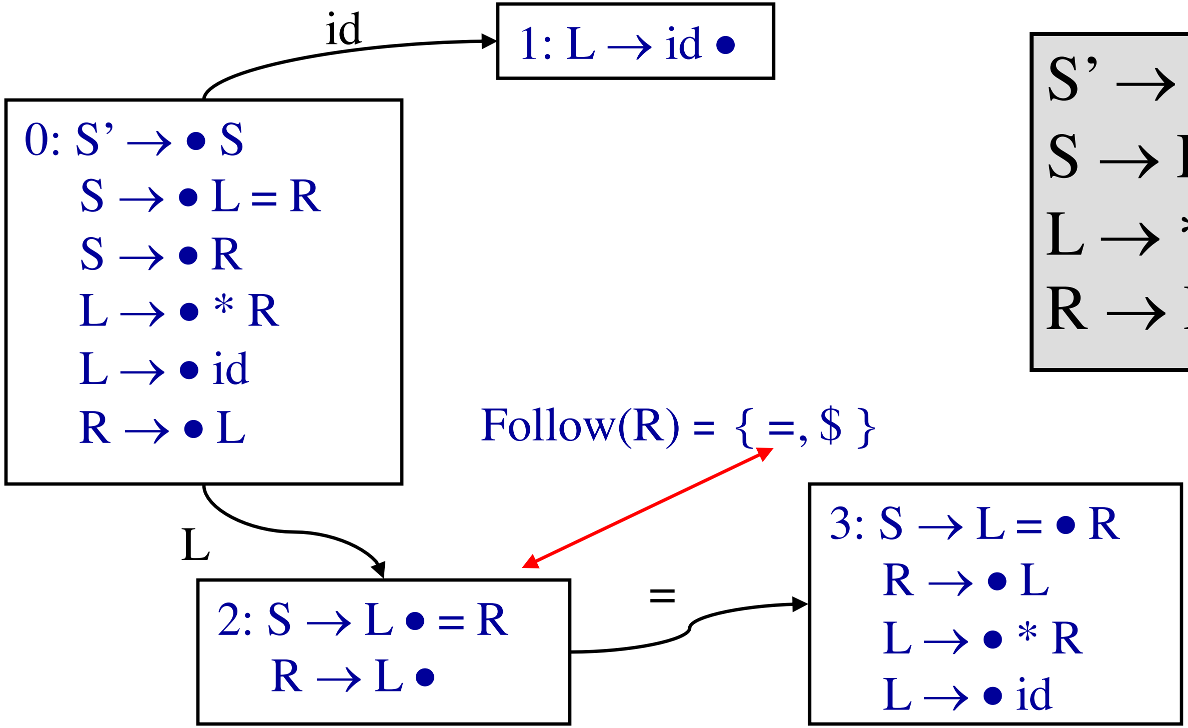
LR Parsing

CMPT 379: Compilers
Instructor: Anoop Sarkar
anoopsarkar.github.io/compilers-class

SLR limitation: lack of context

Input: id = id

S' → S
S → L = R | R
L → *R | id
R → L

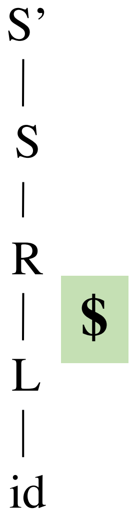


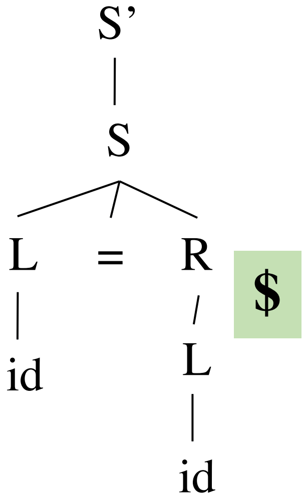
$S' \to S$
$S \to L = R \mid R$
$L \to {*}R \mid id$
$R \to L$

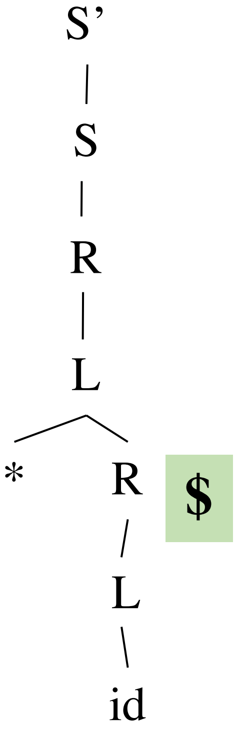
Follow(R) = { =, $ }

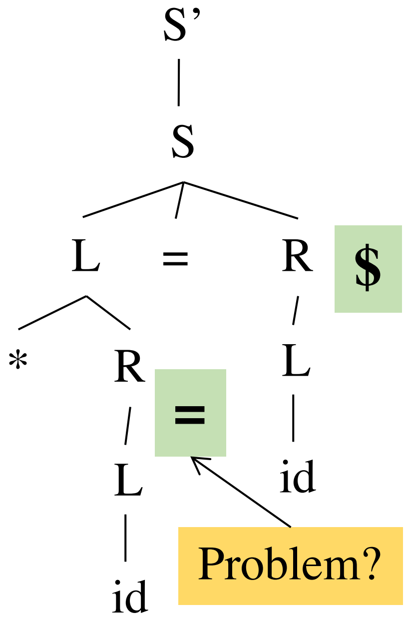
2: $S \to L \bullet = R$
$R \to L \bullet$

Find all lookaheads for reduce $R \to L \bullet$









No! $R \to L \bullet$ reduce and $S \to L \bullet = R$ do not co-occur due to the $L \to {*}R$ rule

Solution: Canonical LR(1)

- Extend definition of configuration
  - Remember lookahead
- New closure method
- Extend definition of Successor

LR(1) Configurations

- $[A \rightarrow \alpha \bullet \beta, a]$ for $a \in T$ is valid for a viable prefix $\delta\alpha$ if there is a rightmost derivation
- $S \Rightarrow^* \delta A \eta \Rightarrow^* \delta \alpha \beta \eta$ and ($\eta = a\gamma$) or ($\eta = \varepsilon$ and $a = \$$)
- Notation: $[A \rightarrow \alpha \bullet \beta, a/b/c]$
  - if $[A \rightarrow \alpha \bullet \beta, a]$, $[A \rightarrow \alpha \bullet \beta, b]$, $[A \rightarrow \alpha \bullet \beta, c]$ are valid configurations

LR(1) Configurations

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

$S \Rightarrow BB \Rightarrow BaB \Rightarrow Bab$
$\Rightarrow aBab \Rightarrow aaBab \Rightarrow aaaBab$

- $S \Rightarrow^*_{rm} aaBab \Rightarrow_{rm} aaaBab$
- Item [B → a • B, a] is valid for **viable prefix** *aaa*
- $S \Rightarrow^*_{rm} BaB \Rightarrow_{rm} BaaB$
- Also, item [B → a • B, $] is valid for viable prefix *Baa*

In $BaB \Rightarrow BaaB$ the string aB is the **handle** (rhs of B)

$S \Rightarrow BB \Rightarrow BaB \Rightarrow BaaB$

LR(1) Closure

Closure property:

- If $[A \rightarrow \alpha \bullet B\beta, a]$ is in set, then $[B \rightarrow \bullet \gamma, b]$ is in set if $b \in \text{First}(\beta a)$
- Compute as fixed point
- Only include contextually valid lookaheads to guide reducing to B

Starting Configuration

- Augment Grammar with S’ just like for LR(0), SLR(1)
- Initial configuration set is

$$I = \text{closure}([S' \rightarrow \bullet\ S, \$])$$

Example: closure([S’ $\rightarrow \bullet$ S, $])

[S’ $\rightarrow \bullet$ S, $]

[S $\rightarrow \bullet$ L = R, $]

[S $\rightarrow \bullet$ R, $]

[L $\rightarrow \bullet$ * R, =]

[L $\rightarrow \bullet$ id, =]

[R $\rightarrow \bullet$ L, $]

[L $\rightarrow \bullet$ *R, $]

[L $\rightarrow \bullet$ id, $]

S’ $\rightarrow$ S
S $\rightarrow$ L = R | R
L $\rightarrow$ *R | id
R $\rightarrow$ L

concisely written as:

S’ $\rightarrow \bullet$ S, $
S $\rightarrow \bullet$ L = R, $
S $\rightarrow \bullet$ R, $
L $\rightarrow \bullet$ * R, =/$
L $\rightarrow \bullet$ id, =/$
R $\rightarrow \bullet$ L, $

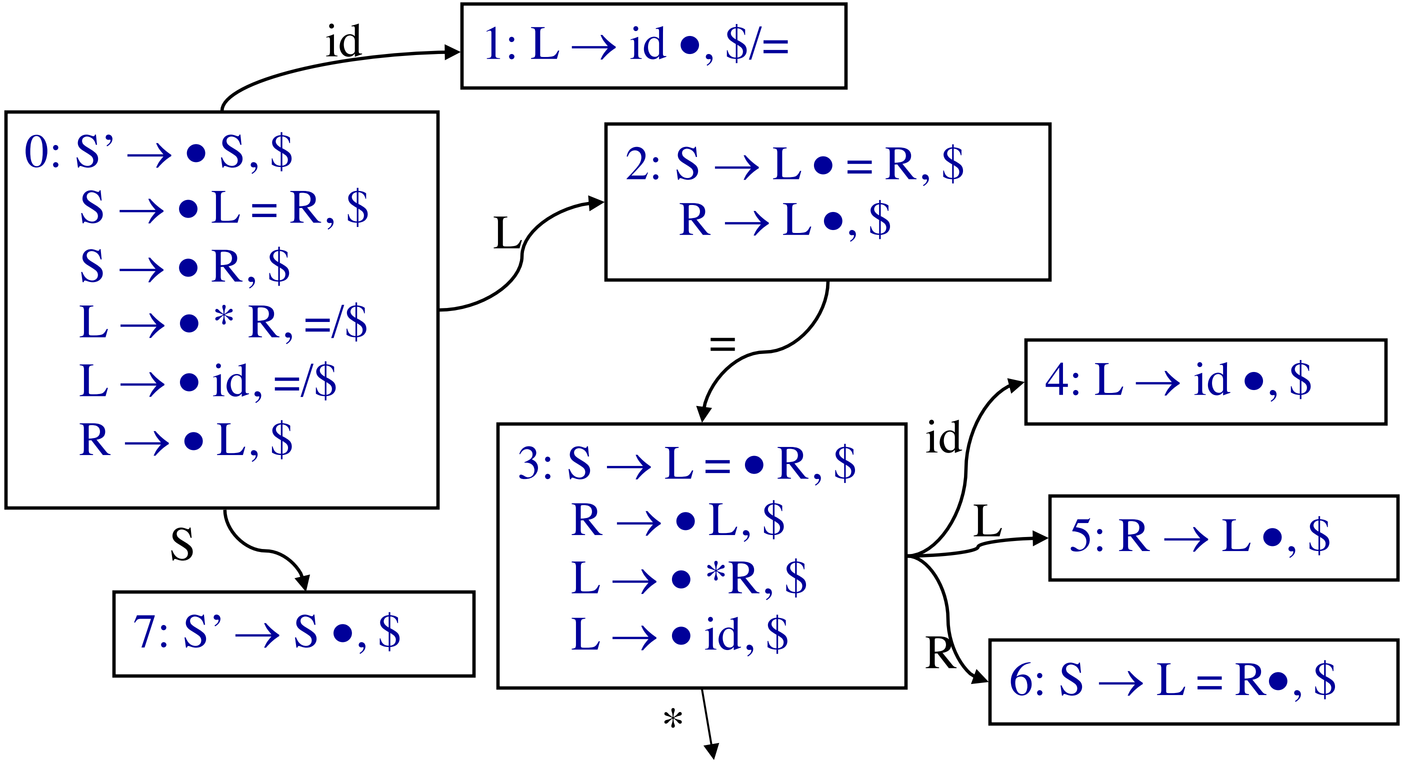
LR(1) Successor(C, X)

- Let I = $[A \rightarrow \alpha \bullet B\beta, a]$ **or** $[A \rightarrow \alpha \bullet b\beta, a]$
- Successor(I, B)

  = closure($[A \rightarrow \alpha B \bullet \beta, a]$)
- Successor(I, b)

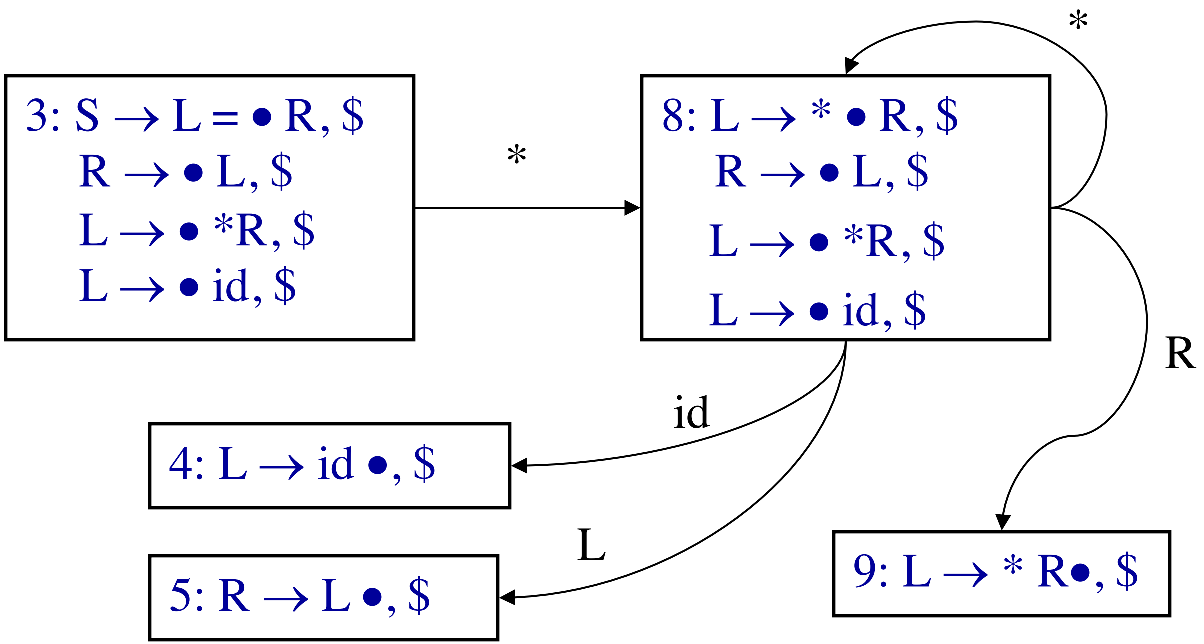
  = closure($[A \rightarrow \alpha b \bullet \beta, a]$)

LR(1) Example

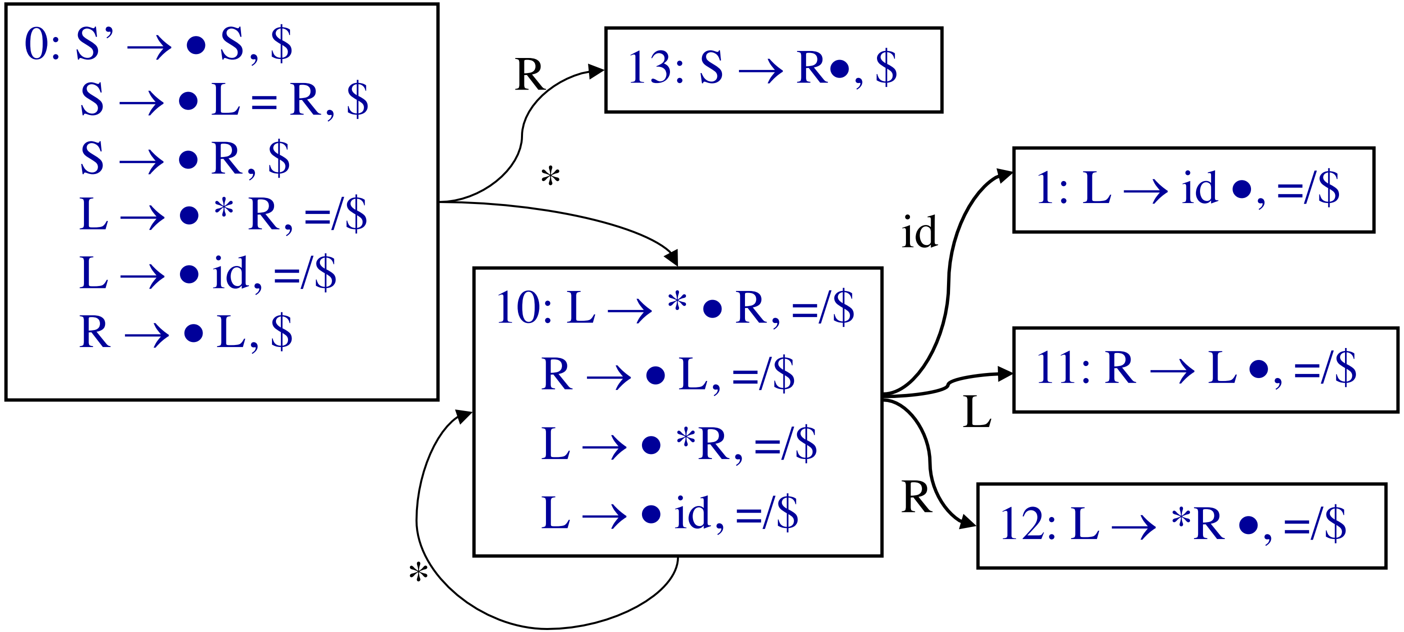


id
1: L → id •, $/=
0: S' → • S, $
S → • L = R, $
S → • R, $
L → • * R, =/$
L → • id, =/$
R → • L, $
L
2: S → L • = R, $
R → L •, $
=
3: S → L = • R, $
R → • L, $
L → • *R, $
L → • id, $
id
4: L → id •, $
L
5: R → L •, $
R
6: S → L = R•, $
S
7: S' → S •, $
*

LR(1) Example (contd)



LR(1) Example (contd)



Productions	
1	S → L = R
2	S → R
3	L → * R
4	L → id
5	R → L

	id	=	*	$	S	L	R
0	S1		S10		7	2	13
1		R4		R4			
2		S3		R5			
3	S4		S8			5	6
4				R4			
5				R5			
6				R1			
7				Acc			
8	S4		S8			5	9
9				R3			
10	S1		S10			11	12
11		R5		R5			
12		R3		R3			
13				R2			

LR(1) Construction

1. Construct $F = \{I_0, I_1, \ldots I_n\}$
2. a) if $[A \rightarrow \alpha\bullet, a] \in I_i$ and $A \mathrel{!}= S'$

   then action[i, a] := reduce $A \rightarrow \alpha$

   b) if $[S' \rightarrow S\bullet, \$] \in I_i$

   then action[i, \$] := accept

   c) if $[A \rightarrow \alpha\bullet a\beta, b] \in I_i$ and $\text{Successor}(I_i, a)=I_j$
   then action[i, a] := shift j
3. if $\text{Successor}(I_i, A) = I_j$ then goto[i, A] := j

LR(1) Construction (cont'd)

$S \rightarrow AaAb$
$S \rightarrow BbBa$
$A \rightarrow \varepsilon$
$B \rightarrow \varepsilon$

Q: Write down the LR(1) automaton and parse table for the above grammar. Is it an LR(1) grammar?

4. All entries not defined are errors
5. Make sure I_0 is the initial state

- Note: LR(1) only reduces using $A \rightarrow \alpha$ for

  $[A \rightarrow \alpha \bullet, a]$ if a is the next input symbol
- LR(1) states remember context by virtue of lookahead
- Possibly many more states than LR(0) due to the lookahead!
  - LALR(1) combines some states

LR(1) Conditions

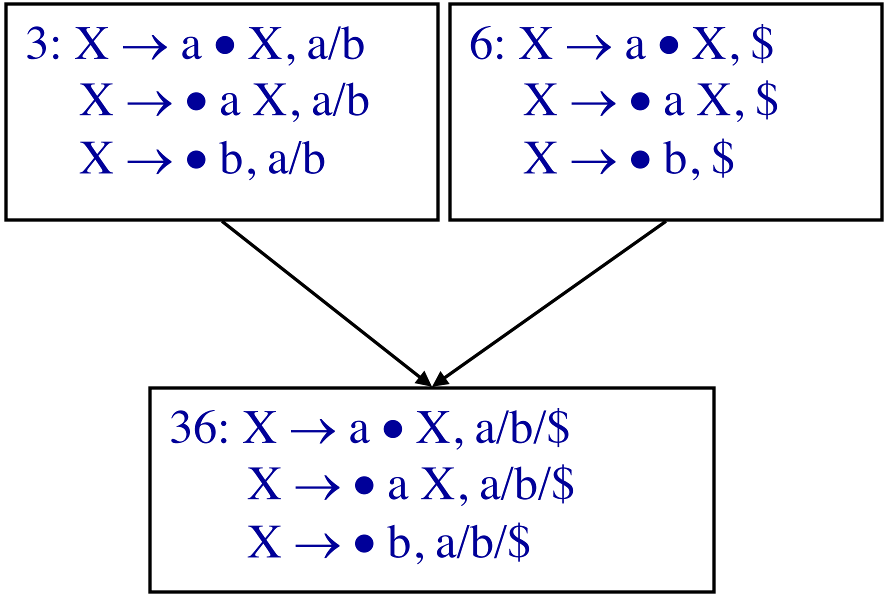
- A grammar is LR(1) if for each configuration set (itemset) the following holds:
  - For any item $[A \rightarrow \alpha \bullet x\beta, a]$ with $x \in T$ there is no $[B \rightarrow \gamma\bullet, x]$
  - For any two complete items $[A \rightarrow \gamma\bullet, a]$ and $[B \rightarrow \beta\bullet, b]$ then a != b.
- Grammars:
  - $LR(0) \subset SLR(1) \subset LR(1) \subset LR(k)$
- Languages expressible by grammars:
  - $LR(0) \subset SLR(1) \subset LR(1) = LR(k)$

Canonical LR(1) Recap

- LR(1) uses left context, current handle and lookahead to decide when to reduce or shift
- Most powerful parser so far (can handle more context-free grammars)
- LALR(1) is practical simplification with fewer states used by yacc/bison to avoid the very large tables generated by LR(1)

Merging States in LALR(1)

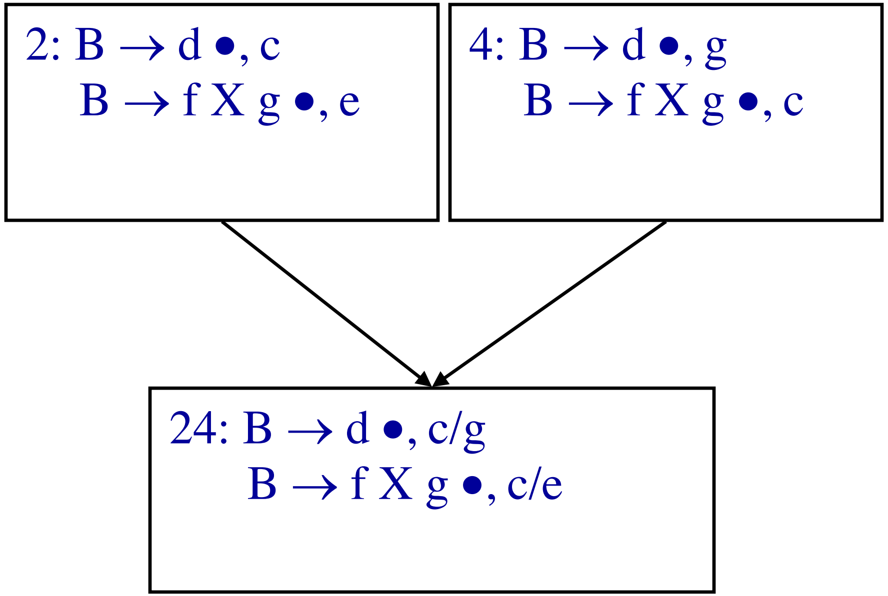
- $S' \rightarrow S$
  $S \rightarrow XX$
  $X \rightarrow aX$
  $X \rightarrow b$
- Same **Core Set**
- Different lookaheads



R/R conflicts when merging

- $B \rightarrow d$
  $B \rightarrow f\ X\ g$
  $X \rightarrow \ldots$

- If R/R conflicts are introduced, grammar is not LALR(1)!



LALR(1)

- LALR(1) Condition:
  - Assumption: merging does not introduce reduce/reduce conflicts
  - Shift/reduce cannot be introduced
- Merging brute force or step-by-step
- More compact than canonical LR, like SLR(1)
- More powerful than SLR(1)
  - Not always merge to full Follow Set

Extra Slides

Set-of-items with Epsilon rules

S → AaAb
S → BbBa
A → ε
B → ε