

## Resolving Conflicts in LALR(1)

- In certain LR(0) states, multiple reductions may be possible.
- Also, reduction and shift might both be possible:
  - Is it OK to take the shift when possible, or is there a problem with the grammar?
- Resolve by computing *lookahead sets* for each item in state that represents possible reduction (dot at end).
- Lookahead set consists of terminal symbols that could legally come next after taking the reduction.
- For item ' $Q \rightarrow a \cdot$ ' in state  $S$ , is set of all terminals that can follow  $Q$  in an input that puts machine in state  $S$ .
- More precise than FOLLOW set from LL(1) parsing.

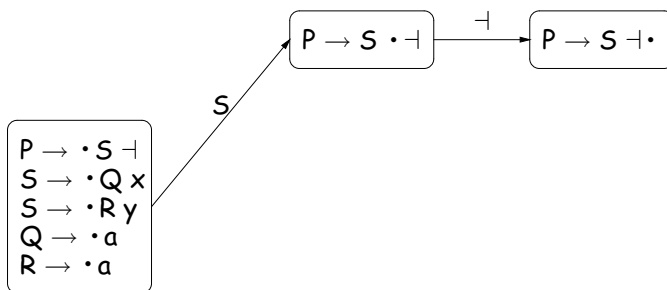
## Finding Lookaheads

- Idea: try possible reductions and see what shifts are then possible.
- Suppose that in state  $S_1$ , two productions  $Q \rightarrow a$  and  $R \rightarrow a$  are both possible.
- Look at what will happen if you reduce with  $Q \rightarrow a$ :
  - Trace backwards from  $S_1$  on the  $a$  arc, and then forward on  $Q$ .
  - Goes to new state  $S_2$ .
- What shifts are possible from  $S_2$ ? Add these to the *lookahead set* for  $Q \rightarrow a$  in  $S_1$ .
- Also consider further reductions from  $S_2$ .
- Do for all states and reduction sequences.
- If lookaheads for  $Q \rightarrow a$  and  $R \rightarrow a$  are distinct, we can decide which production to take.

## Example of LALR(1) Lookaheads

Consider this (silly) grammar and its LR(0) machine (terminals are  $a, x, y, \vdash$ ):

$P \rightarrow S \vdash$   
 $S \rightarrow Q x$   
 $S \rightarrow R y$   
 $Q \rightarrow a$   
 $R \rightarrow a$



## LALR(1) and SLR(1)

- LALR(1) lookahead sets resemble FOLLOW sets.
- In preceding grammar, in fact, could use FOLLOW sets to choose reduction—grammar is SLR(1).
- Not always true. This grammar:

$S \rightarrow A a \mid b A c \mid dc \mid bda$   
 $A \rightarrow d$

has no reduce/reduce conflict, but in the state

$b d \triangleright c$

Must you shift or can you also reduce to  $A$  (meaning there is shift/reduce conflict in the grammar)?

- SLR(1) construction won't tell you. LALR(1) will.