

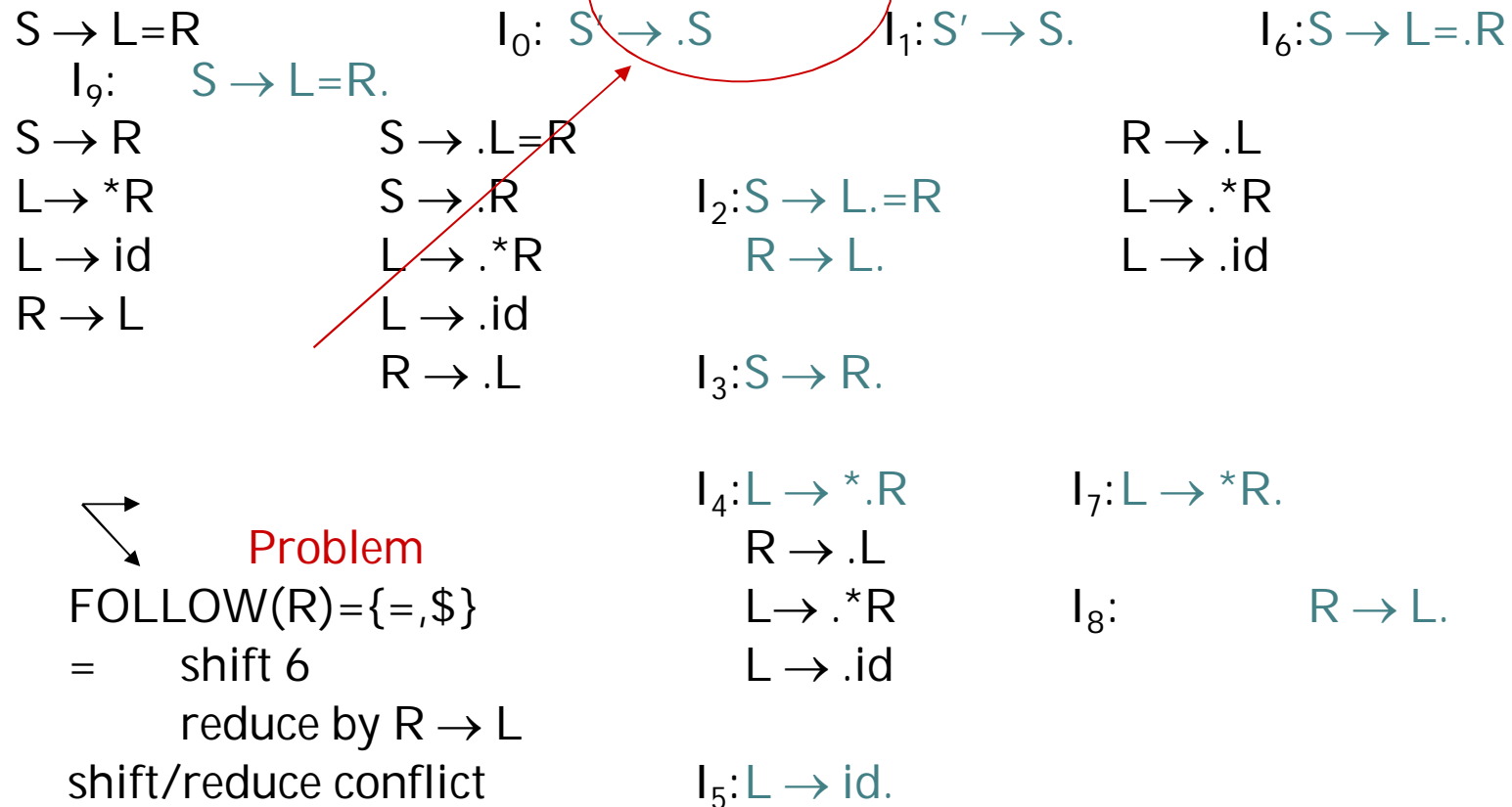
Introduction

LALR Parsing
Constructing Canonical LR(1)
Parsing Tables

LALR Parsing

- Canonical sets of LR(1) items
- Number of states much larger than in the SLR construction
- LR(1) = Order of thousands for a standard prog. Lang.
- SLR(1) = order of hundreds for a standard prog. Lang.
- LALR(1) (lookahead-LR)
- A tradeoff:
 - Collapse states of the LR(1) table that have the same *core* (the “LR(0)” part of each state)
 - LALR never introduces a Shift/Reduce Conflict if LR(1) doesn't.
 - It might introduce a Reduce/Reduce Conflict (that did not exist in the LR(1))...
 - Still much better than SLR(1) (larger set of languages)
 - ... but smaller than LR(1)
- What Yacc and most compilers employ.

Conflict Example



Conflict Example2

$S \rightarrow AaAb$

$S \rightarrow BbBa$

$A \rightarrow \epsilon$

$B \rightarrow \epsilon$

$I_0: S' \rightarrow .S$

$S \rightarrow .AaAb$

$S \rightarrow .BbBa$

$A \rightarrow .$

$B \rightarrow .$

Problem

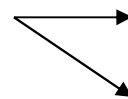
$FOLLOW(A) = \{a, b\}$

$FOLLOW(B) = \{a, b\}$

a reduce by $A \rightarrow \epsilon$

 reduce by $B \rightarrow \epsilon$

reduce/reduce conflict



b reduce by $A \rightarrow \epsilon$

 reduce by $B \rightarrow \epsilon$

reduce/reduce conflict

Constructing Canonical LR(1) Parsing Tables

- In SLR method, the state i makes a reduction by $A \rightarrow \alpha$ when the current token is a :
 - if the $A \rightarrow \alpha \cdot$ in the I_i and a is $\text{FOLLOW}(A)$
- In some situations, βA cannot be followed by the terminal a in a right-sentential form when $\beta \alpha$ and the state i are on the top stack. This means that making reduction in this case is not correct.

$S \rightarrow AaAb$ $S \Rightarrow AaAb \Rightarrow Aab \Rightarrow ab$
 $S \Rightarrow BbBa \Rightarrow Bba \Rightarrow ba$

$S \rightarrow BbBa$

$A \rightarrow \epsilon$

$Aab \Rightarrow \epsilon ab$

$Bba \Rightarrow \epsilon ba$

$B \rightarrow \epsilon$

$AaAb \Rightarrow Aa \epsilon b$

$BbBa \Rightarrow Bb \epsilon a$

LR(1) Item

- To avoid some of invalid reductions, the states need to carry more information.
- Extra information is put into a state by including a terminal symbol as a second component in an item.
- A LR(1) item is:
 $A \rightarrow \alpha.\beta, a$ where **a** is the look-head of
the LR(1) item (**a** is a terminal or end-
marker.)

LR(1) Item (cont.)

- When β (in the LR(1) item $A \rightarrow \alpha.\beta,a$) is not empty, the look-head does not have any affect.
- When β is empty ($A \rightarrow \alpha.,a$), we do the reduction by $A \rightarrow \alpha$ only if the next input symbol is **a** (not for any terminal in FOLLOW(A)).
- A state will contain $A \rightarrow \alpha.,a_1$ where $\{a_1, \dots, a_n\} \subseteq \text{FOLLOW}(A)$

$$\dots$$

$$A \rightarrow \alpha.,a_n$$

Canonical Collection of Sets of LR(1) Items

- The construction of the canonical collection of the sets of LR(1) items are similar to the construction of the canonical collection of the sets of LR(0) items, except that *closure* and *goto* operations work a little bit different.

closure(I) is: (where I is a set of LR(1) items)

- every LR(1) item in I is in closure(I)
- if $A \rightarrow \alpha \cdot B \beta, a$ in closure(I) and $B \rightarrow \gamma$ is a production rule of G; then $B \rightarrow \cdot \gamma, b$ will be in the closure(I) for each terminal b in FIRST(βa) .

goto operation

- If I is a set of LR(1) items and X is a grammar symbol (terminal or non-terminal), then $\text{goto}(I, X)$ is defined as follows:
 - If $A \rightarrow \alpha.X\beta, a$ in I
then every item in **$\text{closure}(\{A \rightarrow \alpha X.\beta, a\})$**
will be in $\text{goto}(I, X)$.