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

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

I_0: $S' \rightarrow .S$
$S \rightarrow .L=R$
$S \rightarrow .R$
$L \rightarrow .*R$
$L \rightarrow .id$
$R \rightarrow .L$

I_1: $S' \rightarrow S.$

I_2: $S \rightarrow L.=R$
$R \rightarrow L.$

I_3: $S \rightarrow R.$

I_4: $L \rightarrow *.R$
$R \rightarrow .L$
$L \rightarrow .*R$
$L \rightarrow .id$

I_5: $L \rightarrow id.$

I_6: $S \rightarrow L=.R$
$R \rightarrow .L$
$L \rightarrow .*R$
$L \rightarrow .id$

I_7: $L \rightarrow *R.$

I_8: $R \rightarrow L.$

I_9: $S \rightarrow L=R.$

Problem
FOLLOW(R)={=,$}
= shift 6
reduce by $R \rightarrow L$
shift/reduce conflict

Conflict Example2

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

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 $A \rightarrow \varepsilon$
reduce by $B \rightarrow \varepsilon$
reduce/reduce conflict

b reduce by $A \rightarrow \varepsilon$
reduce by $B \rightarrow \varepsilon$
reduce/reduce conflict

Constructing Canonical LR(1) Parsing Tables

- In SLR method, the state i makes a reduction by $A \to \alpha$ when the current token is a:
  - if the $A \to \alpha .$ in the I_i and a is 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.

$S \to AaAb$ $\quad S \Rightarrow AaAb \Rightarrow Aab \Rightarrow ab$

$S \Rightarrow BbBa \Rightarrow Bba \Rightarrow ba$

$S \to BbBa$

$A \to \varepsilon$ $\quad Aab \Rightarrow \varepsilon\, ab$ $\quad Bba \Rightarrow \varepsilon\, ba$

$B \to \varepsilon$ $\quad AaAb \Rightarrow Aa\, \varepsilon\, b$ $\quad BbBa \Rightarrow Bb\, \varepsilon\, a$

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 $\{a_1,...,a_n\} \subseteq$ FOLLOW(A)

$$\ldots$$

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

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 $A \rightarrow \alpha . B\beta, a$ in closure(I) and $B \rightarrow \gamma$ is a production rule of G; then $B \rightarrow .\gamma, b$ 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 goto(I,X) is defined as follows:
  - If $A \rightarrow \alpha.X\beta,a$ in I then every item in **closure({$A \rightarrow \alpha X.\beta,a$})** will be in goto(I,X).