

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

$x=y: \quad \begin{array}{c} S \\ / \quad \backslash \\ L \quad R \\ / \quad \backslash \\ id \quad id \end{array}$

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

$I_1 = trans(I_0, L): \quad S \rightarrow L = R$
 $R \rightarrow L$

$action[I_1, =] = shift \quad I_1 \rightarrow S \rightarrow L = R$

$Follow(S) = \{ =, *, id \}$
 $Follow(R) = \{ =, id \}$

Shift/reduce conflict!
 When resolves shift/reduce conflict by shifting (by default)
 When resolves a reduce/reduce conflict by reducing by the product occurring earlier in the rules section (by default)
 Change defaults using $\%prec$ and $\%assoc$ directives
 Canonical LR parsing (LR(1))

Apr 18-11:00 AM

An LR(1) item is of the form
 $[A \rightarrow \alpha \cdot B \mid a]$

where $A \rightarrow \alpha \beta$ is a production and a is either a terminal or $\$$.

This item "means", we are currently working on production $A \rightarrow \alpha \beta$, having parsed α already but not β , and we expect to see a as the lookahead when this production finishes (reduction).

Let S be some set of LR(1) items (i.e., a state)
 closure(S)
 repeat
 for each item $[A \rightarrow \alpha \cdot B \mid a]$ in S , do
 for each production $B \rightarrow \gamma$ of the grammar, do
 for each $b \in Follow(B)$
 add $[B \rightarrow \gamma \cdot b]$ to S
 until nothing more is added to S

Apr 18-11:21 AM

$trans(S, X):$
 $t = \emptyset$
 For each item $[A \rightarrow \alpha \cdot X \beta \mid a] \in S$
 add $[A \rightarrow \alpha X \cdot \beta \mid a]$ to t
 return closure(t).

Start state is
 $I_0 := closure(\{[S' \rightarrow S \cdot]\})$

Action table:
 Set $action(I, \cdot) := \text{empty}$
 where $I_1 = trans(I_0, S)$
 $= \{[S' \rightarrow S \cdot]\}$

For any state I and terminal a, β
 $[A \rightarrow \alpha \cdot a \beta \mid b] \in I$
 (for some A, α, β, b), then
 set $action[I, a] =$
 shift to J where
 $J := trans(I, a)$

If $[A \rightarrow \alpha \cdot a] \in I$
 then set
 $action[I, a] := \text{reduce } A \rightarrow \alpha$

Apr 18-11:32 AM

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

$I_0: S' \rightarrow S, \$$
 $S \rightarrow L, \$$
 $S \rightarrow R, \$$
 $L \rightarrow *, \$$
 $L \rightarrow id, \$$
 $R \rightarrow L, \$$

$I_1 = trans(I_0, S): [S' \rightarrow S \cdot, \$]$

$I_2 = trans(I_1, L):$
 $S \rightarrow L = R, \$$
 $R \rightarrow L, \$$

$trans(I_1, =):$
 $S \rightarrow L = R, \$$
 $R \rightarrow L, \$$

Suppose (hypothetical) closure: if
 $[A \rightarrow \alpha \cdot B \beta \mid a]$
 is in the state and
 $B \rightarrow \gamma$ is production,
 add $[B \rightarrow \gamma \cdot b]$ for
 every $b \in Follow(B)$
 This is no better than SLR parsing!
 Canonical LR(1) is more powerful than SLR(1)

Apr 18-11:43 AM

Problem: many more LR(1) items than LR(0) items.
 get a huge blowup in the number of states;
 parser is too big to be practical.

Compromise: LALR parser
 states are sets of LR(1) items, but condense several states into one. Usually get a powerful (enough) parser that is still reasonably compact

This is LALR (produces an LALR parser)
 (Defer until Wednesday)

Dangling "else" ambiguity.
 $S \rightarrow if\ e\ S$
 $S \rightarrow if\ e\ S\ else\ S$
 $S \rightarrow \text{other}$

Input:
 $if\ e\ if\ e\ other\ else\ other$

Apr 18-11:56 AM

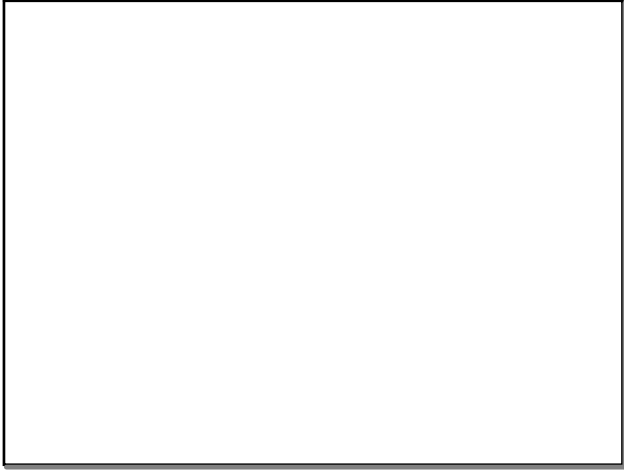
Some state I contains both
 $[S \rightarrow if\ e\ S \cdot, \text{else}]$
 and
 $[S \rightarrow if\ e\ S \cdot \text{else } S, \dots]$

$action[I, \text{else}]:$ shift ---
 reduce

bison's default behavior (shifting) chooses the correct parse tree in this case. So this particular conflict is tolerated

Ex: (1) Check that shifting is the right thing really.
 (2) Rewrite the grammar to be unambiguous.

Apr 18-12:10 PM



Apr 18-12:12 PM