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 $A \rightarrow \alpha.$ 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.

$S \rightarrow AaAb$	$S \Rightarrow AaAb \Rightarrow Aab \Rightarrow ab$	$S \Rightarrow BbBa \Rightarrow Bba \Rightarrow ba$
$S \rightarrow BbBa$		
$A \rightarrow \varepsilon$	$Aab \Rightarrow \varepsilon\ ab$	$Bba \Rightarrow \varepsilon\ ba$
$B \rightarrow \varepsilon$	$AaAb \Rightarrow Aa\ \varepsilon\ b$	$BbBa \Rightarrow Bb\ \varepsilon\ a$

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:

  $A \to \alpha \bullet \beta, a$ 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 $A \rightarrow \alpha\bullet\beta,a$) is not empty, the look-head does not have any affect.
- When β is empty ($A \rightarrow \alpha\bullet,a$), 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 $A \rightarrow \alpha\bullet,a_1$ where $\{a_1,...,a_n\} \subseteq$ FOLLOW(A)

  ...

  $A \rightarrow \alpha\bullet,a_n$

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 $A \rightarrow \alpha \bullet B\beta, a$ 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).

Construction of The Canonical LR(1) Collection

- ***Algorithm***:

  C is { closure({S'→.S,$}) }

  repeat the followings until no more set of LR(1) items can be added to ***C***.

      for each I in ***C*** and each grammar symbol X

          if goto(I,X) is not empty and not in ***C***

              add goto(I,X) to ***C***

- goto function is a DFA on the sets in C.

A Short Notation for The Sets of LR(1) Items

- A set of LR(1) items containing the following items

$$A \rightarrow \alpha \bullet \beta, a_1$$

$$\ldots$$

$$A \rightarrow \alpha \bullet \beta, a_n$$

can be written as

$$A \rightarrow \alpha \bullet \beta, a_1/a_2/\ldots/a_n$$

Canonical LR(1) Collection -- Example

$S \rightarrow AaAb$

$S \rightarrow BbBa$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

I_0: $S' \rightarrow .S ,\$$
$S \rightarrow .AaAb ,\$$
$S \rightarrow .BbBa ,\$$
$A \rightarrow . ,a$
$B \rightarrow . ,b$

$I_0 \xrightarrow{S} I_1$: $S' \rightarrow S. ,\$$

$I_0 \xrightarrow{A} I_2$: $S \rightarrow A.aAb ,\$ \xrightarrow{a}$ to I_4

$I_0 \xrightarrow{B} I_3$: $S \rightarrow B.bBa ,\$ \xrightarrow{b}$ to I_5

I_4: $S \rightarrow Aa.Ab ,\$ \xrightarrow{A}$ I_6: $S \rightarrow AaA.b ,\$ \xrightarrow{a}$ I_8: $S \rightarrow AaAb. ,\$$
$A \rightarrow . ,b$

I_5: $S \rightarrow Bb.Ba ,\$ \xrightarrow{B}$ I_7: $S \rightarrow BbB.a ,\$ \xrightarrow{b}$ I_9: $S \rightarrow BbBa. ,\$$
$B \rightarrow . ,a$

Canonical LR(1) Collection – Example2

$S' \to S$
1) $S \to L{=}R$
2) $S \to R$
3) $L \to {*}R$
4) $L \to id$
5) $R \to L$

I_0: $S' \to .S,\$$
$S \to .L{=}R,\$$
$S \to .R,\$$
$L \to .{*}R,\$/{=}$
$L \to .id,\$/{=}$
$R \to .L,\$$

$I_0 \xrightarrow{S} I_1$, $I_0 \xrightarrow{*} I_4$, $I_0 \xrightarrow{L} I_2$, $I_0 \xrightarrow{R} I_3$, $I_0 \xrightarrow{id} I_5$

I_1: $S' \to S.,\$$

I_2: $S \to L.{=}R,\$$ $\longrightarrow$ to I_6
$R \to L.,\$$

I_3: $S \to R.,\$$

I_4: $L \to {*}.R,\$/{=}$
$R \to .L,\$/{=}$
$L \to .{*}R,\$/{=}$
$L \to .id,\$/{=}$

$\xrightarrow{R}$ to I_7
$\xrightarrow{L}$ to I_8
$\xrightarrow{*}$ to I_4
$\xrightarrow{id}$ to I_5

I_5: $L \to id.,\$/{=}$

I_6: $S \to L{=}.R,\$$
$R \to .L,\$$
$L \to .{*}R,\$$
$L \to .id,\$$

$\xrightarrow{R}$ to I_9
$\xrightarrow{L}$ to I_{10}
$\xrightarrow{*}$ to I_{11}
$\xrightarrow{id}$ to I_{12}

I_7: $L \to {*}R.,\$/{=}$

I_8: $R \to L.,\$/{=}$

I_9: $S \to L{=}R.,\$$

I_{10}: $R \to L.,\$$

I_{11}: $L \to {*}.R,\$$
$R \to .L,\$$
$L \to .{*}R,\$$
$L \to .id,\$$

$\xrightarrow{R}$ to I_{13}
$\xrightarrow{L}$ to I_{10}
$\xrightarrow{*}$ to I_{11}
$\xrightarrow{id}$ to I_{12}

I_{12}: $L \to id.,\$$

I_{13}: $L \to {*}R.,\$$

I_4 and I_{11}

I_5 and I_{12}

I_7 and I_{13}

I_8 and I_{10}

Construction of LR(1) Parsing Tables

1. Construct the canonical collection of sets of LR(1) items for G'.
   $C \leftarrow \{I_0,...,I_n\}$

2. Create the parsing action table as follows
   - If a is a terminal, $A \rightarrow \alpha \bullet a\beta, b$ in I_i and $goto(I_i,a)=I_j$ then action[i,a] is ***shift j.***
   - If $A \rightarrow \alpha \bullet, a$ is in I_i, then action[i,a] is ***reduce*** $\boldsymbol{A \rightarrow \alpha}$ where $A \neq S'$.
   - If $S' \rightarrow S \bullet, \$$ is in I_i, then action[i,$] is ***accept***.
   - If any conflicting actions generated by these rules, the grammar is not LR(1).

3. Create the parsing goto table
   - for all non-terminals A, if $goto(I_i,A)=I_j$ then goto[i,A]=j

4. All entries not defined by (2) and (3) are errors.

5. Initial state of the parser contains $S' \rightarrow .S, \$$

LR(1) Parsing Tables – (for Example2)

	id	*	=	$	S	L	R
0	s5	s4			1	2	3
1				acc			
2			s6	r5			
3				r2			
4	s5	s4				8	7
5			r4	r4			
6	s12	s11				10	9
7			r3	r3			
8			r5	r5			
9				r1			
10				r5			
11	s12	s11				10	13
12				r4			
13				r3			

no shift/reduce or
no reduce/reduce conflict

$\Downarrow$

so, it is a LR(1) grammar