UCS1602:
COMPILER DESIGN

Canonical LR

Session Outcomes

- At the end of this session, participants will be able to
  - Understand the concepts of CLR parser
  - Design CLR parser

Outline

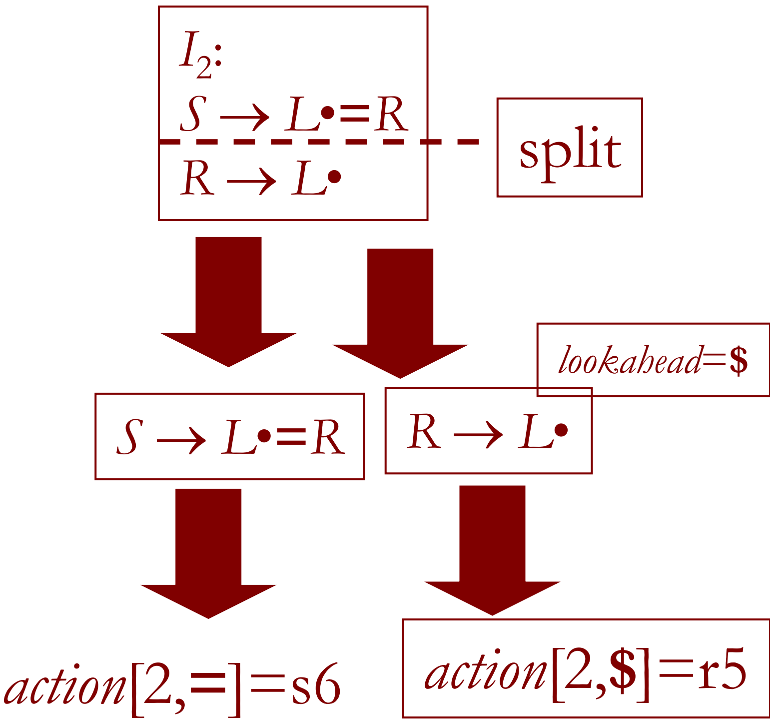
- CLR Parser
- Augmented grammar
- LR(1) Item construction
- CLR parsing table construction
- LR parsing algorithm

Introduction

- 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.

Introduction

- Split the SLR states by adding LR(1) lookahead
- Unambiguous grammar
  1. $S \rightarrow L = R$
  2. $S \rightarrow R$
  3. $L \rightarrow * R$
  4. $L \rightarrow \mathbf{id}$
  5. $R \rightarrow L$



Should not reduce on **=**, because no right-sentential form begins with *R*=

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).

Construction of 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.

Short Notation for Sets of LR(1) Items

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

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

$$\ldots$$

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

can be written as

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

Example LR(1) Items

- Unambiguous LR(1) grammar:

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

  $S \rightarrow R$

  $L \rightarrow$ ***** R

  $L \rightarrow$ **id**

  $R \rightarrow L$
- Augment with $S' \rightarrow S$
- LR(1) items (next slide)

LR(1) Item

I_0: [$S' \to \bullet S$,**$**] goto($I_0$,$S$)=$I_1$
[$S \to \bullet L$**=**R,**$**] goto($I_0$,$L$)=$I_2$
[$S \to \bullet R$,**$**] goto($I_0$,$R$)=$I_3$
[$L \to \bullet$*****R,**= / $**] goto($I_0$,*****)=$I_4$
[$L \to \bullet$**id**,**=/ $**] goto($I_0$,**id**)=$I_5$
[$R \to \bullet L$,**$**] goto($I_0$,$L$)=$I_2$

I_1: [$S' \to S\bullet$,**$**]

I_2: [$S \to L\bullet$**=**R,**$**] goto($I_0$,**=**)=$I_6$
[$R \to L\bullet$,**$**]

I_3: [$S \to R\bullet$,**$**]

I_4: [$L \to$ *****$\bullet R$,**= / $**] goto($I4$,$R$)=$I7$
[$R \to \bullet L$,**=/ $**] goto($I4$,$L$)=$I8$
[$L \to \bullet$*****R,**= / $**] goto($I4$,*****)=$I4$
[$L \to \bullet$**id**,**= / $**] goto($I4$,**id**)=$I5$

I_5: [$L \to$ **id**$\bullet$,**= / $**]

Canonical LR(1) Collection

$S' \to S$
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: $S \to L.=R,\$$ $\to$ 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_{13}: $L \to *R.,\$$

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_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_7: $L \to *R.,=/\$$

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

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

I_4 and I_{11}

I_5 and I_{12}

I_7 and I_{13}

I_8 and I_{10}

Canonical LR Parsing Tables

1. Augment the grammar with $S' \rightarrow S$
2. Construct the set $C=\{I_0, I_1, \ldots, I_n\}$ of LR(1) items
3. If $[A \rightarrow \alpha \bullet a \beta,\ b] \in I_i$ and $goto(I_i, a) = I_j$ then set $action[i,a]$=shift j
4. If $[A \rightarrow \alpha \bullet,\ a] \in I_i$ then set $action[i,a]$=reduce $A \rightarrow \alpha$ (apply only if $A \neq S'$)
5. If $[S' \rightarrow S \bullet,\ \$]$ is in I_i then set $action[i,\$]$=accept
6. If $goto(I_i, A) = I_j$ then set $goto[i,A] = j$
7. Repeat 3-6 until no more entries added
8. The initial state i is the I_i holding item $[S' \rightarrow \bullet S, \$]$

Parsing Table

Grammar:

1. $S \rightarrow L = R$
2. $S \rightarrow R$
3. $L \rightarrow * R$
4. $L \rightarrow \textbf{id}$
5. $R \rightarrow L$

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

Summary

- LR Parsers
- Augmented grammar
- LR(1) Item construction
- CLR parsing table construction

Check your understanding?

1. Consider the following grammar

   S → CC

   C → cC

   C→ d

Construct CLR parsing table for the above grammar. Parse the input cdcd