Tabelle LR

Costruzione delle tabelle di parsing LR canoniche

Maria Rita Di Berardini

Dipartimento di Matematica e Informatica
Universitá di Camerino
mariarita.diberardini@unicam.it

Grammatiche non SLR

- Grammatiche ambigue non possono essere LR, e quindi SLR(1)
- Tuttavia ci sono grammatiche non ambigue che non sono SLR(1)
- Consideriamo la seguente grammatica (aumentata) G'

$$
\begin{aligned}
S' &\rightarrow S \\
S &\rightarrow L = R \\
S &\rightarrow R \\
L &\rightarrow *R \\
L &\rightarrow \mathbf{id} \\
R &\rightarrow L
\end{aligned}
$$

è non ambigua ma nemmeno SLR(1)

Una grammatica non SLR(1)

- Costruiamo la collezione canonica di item LR(0) per G':

$$
\begin{aligned}
S' &\to S \\
S &\to L = R \\
S &\to R \\
L &\to *R \\
L &\to \mathbf{id} \\
R &\to L
\end{aligned}
$$

- $I_0 = \{S' \to \bullet S, S \to \bullet L = R, S \to \bullet R, L \to \bullet * R, L \to \bullet \mathbf{id}, R \to \bullet L\}$
- $I_1 = goto(I_0, S) = \{S' \to S\bullet\}$
- $I_2 = goto(I_0, L) = \{S \to L\bullet = R, R \to L\bullet\}$
- $I_3 = goto(I_0, R) = \{S \to R\bullet\}$
- $I_4 = goto(I_0, *) = \{L \to * \bullet R, R \to \bullet L, L \to \bullet * R, L \to \bullet \mathbf{id}\}$
- $I_5 = goto(I_0, id) = \{L \to \mathbf{id}\bullet\}$

Una grammatica non SLR(1)

- Costruiamo la collezione canonica di item LR(0) per G':

$$S' \rightarrow S$$
$$S \rightarrow L = R$$
$$S \rightarrow R$$
$$L \rightarrow *R$$
$$L \rightarrow \mathbf{id}$$
$$R \rightarrow L$$

- $I_6 = goto(I_2, =)\{S \rightarrow L = \bullet R, R \rightarrow \bullet L, L \rightarrow \bullet * R, L \rightarrow \bullet \mathbf{id}\}$
- $I_7 = goto(I_4, R) = \{L \rightarrow *R\bullet\}$
- $I_8 = goto(I_4, L) = \{R \rightarrow L\bullet\}$
- $goto(I_4, *) = I_4$, $goto(I_4, id) = I_5$
- $goto(I_6, R) = \{S \rightarrow L = R\bullet\} = I_9$
- $goto(I_6, L) = I_8$, $goto(I_6, *) = I_4$, $goto(I_6, \mathbf{id}) = I_5$

Una grammatica non SLR(1)

- Analizziamo l'insieme $I_2 = goto(I_0, L) = \{S \rightarrow L\bullet = R, R \rightarrow L\bullet\}$
- Poichè $goto(I_2, =) = I_6$, possiamo aggiungere shifts_6 nella entry $action[s_2, =]$

Una grammatica non SLR(1)

- Analizziamo l'insieme $I_2 = goto(I_0, L) = \{S \rightarrow L\bullet = R, R \rightarrow L\bullet\}$
- Poichè $goto(I_2, =) = I_6$, possiamo aggiungere shifts_6 nella entry $action[s_2, =]$
- Inoltre, $R \rightarrow L\bullet \in I_2$ e FOLLOW$(R) = \{\$, =\}$ implicano $action[s_2, =] = action[s_2, \$] =$ reduce $R \rightarrow L$
- Abbiamo un conflitto shift/reduce in $action[s_2, =]$
- Il parser non ha abbastanza informazioni per decidere cosa fare se, nello stato s_2, legge il simbolo $=$

La tabella SLR(1)

	id	+	*	=	$	S	L	R
s_0	s5		s4			1	2	3
s_1					acc			
s_2				s6/r5	r5			
s_3					r2			
s_4	s5		s4				8	7
s_5				r4	r4			
s_6	s5		s4				8	9
s_7				r3	r3			
s_8				r5	r5			
s_9					r1			

Parsing della stringa $*\mathbf{id} = \mathbf{id}$

	STACK	INPUT	AZIONE
1	s_0	$*\mathbf{id} = \mathbf{id}\$$	shift 4
2	$s_0 * s_4$	$\mathbf{id} = \mathbf{id}\$$	shift 5
3	$s_0 * s_4 \mathbf{id} s_5$	$= \mathbf{id}\$$	reduce $L \rightarrow \mathbf{id}$
4	$s_0 * s_4 L s_8$	$= \mathbf{id}\$$	reduce $R \rightarrow L$
5	$s_0 * s_4 R s_7$	$= \mathbf{id}\$$	reduce $L \rightarrow *R$
6	$s_0 L s_2$	$= \mathbf{id}\$$	assumiamo di shiftare allo stato s_6
7	$s_0 L s_2 = s_6$	$\mathbf{id}\$$	shifts_5
8	$s_0 L s_2 = s_6 \mathbf{id} s_5$	$\$$	reduce$L \rightarrow \mathbf{id}$
9	$s_0 L s_2 = s_6 L s_8$	$\$$	reduce$R \rightarrow L$
10	$s_0 L s_2 = s_6 R s_9$	$\$$	reduce$S \rightarrow L = R$
11	$s_0 S s_1$	$\$$	acc

Parsing della stringa $*\mathbf{id} = \mathbf{id}$

STEP	STACK	INPUT	AZIONE
1	s_0	$*\mathbf{id} = \mathbf{id}\$$	shift 4
2	$s_0 * s_4$	$\mathbf{id} = \mathbf{id}\$$	shift 5
3	$s_0 * s_4 \mathbf{id} s_5$	$= \mathbf{id}\$$	reduce $L \rightarrow \mathbf{id}$
4	$s_0 * s_4 L s_8$	$= \mathbf{id}\$$	reduce $R \rightarrow L$
5	$s_0 * s_4 R s_7$	$= \mathbf{id}\$$	reduce $L \rightarrow *R$
6	$s_0 L s_2$	$= \mathbf{id}\$$	s6/r5 assumiamo di ridurre con $R \rightarrow L$
6	$s_0 R s_3$	$= \mathbf{id}\$$	errore

Il prossimo handle dovrebbe cominciare con **R=** il che non è possibile viste le produzione della nostra grammatica

Soluzione: aggiungiamo informazione agli stati

- Ogni stato deve contenere degli item LR(0) ma anche dei simboli di input rispetto ai quali ridurre nel momento in cui al top dello stack si forma l'handle
- Gli item diventani delle coppie $[A \rightarrow \alpha \bullet \beta, a]$ dove:
  - $A \rightarrow \alpha \bullet \beta$ è un "vecchio" item LR(0)
  - a è il **simbolo di lookahead**: un terminale oppure il \$
- Questi items vengono detti item LR(1)

Soluzione: aggiungiamo informazione agli stati

- Ogni stato deve contenere degli item LR(0) ma anche dei simboli di input rispetto ai quali ridurre nel momento in cui al top dello stack si forma l'handle
- Gli item diventani delle coppie $[A \rightarrow \alpha \bullet \beta, a]$ dove:
  - $A \rightarrow \alpha \bullet \beta$ è un "vecchio" item LR(0)
  - a è il **simbolo di lookahead**: un terminale oppure il \$
- Questi items vengono detti item LR(1)
- Il simbolo di lookahead non ha effetti su items del tipo $[A \rightarrow \alpha \bullet \beta, a]$ con $\beta \neq \epsilon$, ma

Soluzione: aggiungiamo informazione agli stati

- Ogni stato deve contenere degli item LR(0) ma anche dei simboli di input rispetto ai quali ridurre nel momento in cui al top dello stack si forma l'handle
- Gli item diventani delle coppie $[A \rightarrow \alpha \bullet \beta, a]$ dove:
  - $A \rightarrow \alpha \bullet \beta$ è un "vecchio" item LR(0)
  - a è il **simbolo di lookahead**: un terminale oppure il \$
- Questi items vengono detti item LR(1)
- Il simbolo di lookahead non ha effetti su items del tipo $[A \rightarrow \alpha \bullet \beta, a]$ con $\beta \neq \epsilon$, ma
- per ogni l'item è della forma $[A \rightarrow \alpha \bullet, a]$ allora *si esegue una riduzione solo se il corrente simbolo in input* è a
- Non riduciamo per tutti i simboli in FOLLOW(A), ma per un sottoinsieme (proprio) di questi

Costruzione della collezione canonica

- La costruzione della collezione canonica di item LR(1) è la stessa vista per la collezione canonica LR(0)
- La funzione *goto* è simile a quella vista per insiemi di items LR(0).

function $goto(I, X)$;
begin
Sia $J = \{[A \rightarrow \alpha X \bullet \beta, a] \mid [A \rightarrow \alpha \bullet X \beta, a] \in I\}$
return $closure(J)$;
end;

- L'unica modifica sostanziale è quella della funzione *closure*

La *closure* per insiemi di items LR(1)

begin
closure(I) := I
repeat
for each ($[A \to \alpha \bullet B\beta, a] \in I$, $B \to \gamma \in G'$, and $b \in \text{FIRST}(\beta a)$)
do aggiungi $[B \to \bullet\gamma, b]$ in *closure*(I)
until (nessun nuovo item può essere aggiunto)
end;

Calcolo della *closure*(I): un esempio

$S' \to S$
$S \to CC$
$C \to cC \mid d$

e calcoliamo $I_0 = closure(\{[S' \to \bullet S, \$]\})$

- Inanzittuo, aggiungiamo l'item $[S' \to \bullet S, \$]$.

Calcolo della *closure*(I): un esempio

$S' \to S$
$S \to CC$
$C \to cC \mid d$

e calcoliamo $I_0 = \mathit{closure}(\{[S' \to \bullet S, \$]\})$

- Inanzittuo, aggiungiamo l'item $[S' \to \bullet S, \$]$.
- Poichè FIRST(βa) = FIRST($\$a$) = $\{\$\}$, aggiungiamo anche $[S \to \bullet CC, \$]$

Calcolo della *closure*(I): un esempio

$S' \to S$
$S \to CC$
$C \to cC \mid d$

e calcoliamo $I_0 = closure(\{[S' \to \bullet S, \$]\})$

- Inanzittuo, aggiungiamo l'item $[S' \to \bullet S, \$]$.
- Poichè FIRST(βa) = FIRST$(\$a) = \{\$\}$, aggiungiamo anche $[S \to \bullet CC, \$]$
- Poichè FIRST(βa) = FIRST$(C\$)$ = FIRST$(C) = \{c, d\}$, aggiungiamo gli items:

$$[C \to \bullet cC, c],\ [C \to \bullet cC, d]$$
$$[C \to \bullet d, c],\ [C \to \bullet d, d]$$

Calcolo della *closure*(I): un esempio

Ricapitolando:

$$
\begin{aligned}
I_0 \quad = \{ & \\
& [S' \rightarrow \bullet S, \$], \\
& [S \rightarrow \bullet CC, \$] \\
& [C \rightarrow \bullet cC, c], \\
& [C \rightarrow \bullet cC, d] \\
& [C \rightarrow \bullet d, c], \\
& [C \rightarrow \bullet d, d] \\
\} &
\end{aligned}
$$

Abbreviazioni

$$
\begin{aligned}
& [S' \rightarrow \bullet S, \$] \\
& [S \rightarrow \bullet CC, \$], \\
& [C \rightarrow \bullet cC, c/d], \\
& [C \rightarrow \bullet d, c/d]
\end{aligned}
$$

La funzione *goto*

function *goto*(I, X);
dove I è un insieme di item LR(1) ed X è un simbolo della gram.
begin
Sia J l'insieme di item LR(1) della forma $[A \rightarrow \alpha X \bullet \beta, a]$ tali che $[A \rightarrow \alpha \bullet X\beta, a] \in I$
return *closure*(J);
end;

Calcolo degli Items LR(1)

- Utilizziamo la stessa identica procedura che abbiamo visto per la collezione canonica LR(0), ma con le funzioni *closure* e *goto* nuove
- Anche in questo caso il risultato è un automa deterministico i cui stati sono insiemi di item LR(1)
- Continuiamo con l'esempio

Calcolo della *closure*(I): un esempio

$goto(I_0, S) = closure(\{[S' \to S\bullet, \$]\}) = \{[S' \to S\bullet, \$]\} = I_1$

$goto(I_0, C) = closure(\{[S \to C \bullet C, \$]\})$

La $closure(\{[S' \to S\bullet, \$]\})$ contiene, inanzitutto, l'item $[S \to C \bullet C, \$]$. Questo item matcha con il template $[A \to \alpha \bullet B\beta, a]$ ponendo $A = S$, $\alpha = C$, $B = C$, $\beta = \varepsilon$ e $a = \$$ (e quindi FIRST$(\beta\, a) = \{\$\}$)

Aggiungiamo i seguenti items: $[C \to \bullet cC, \$]$ e $[C \to \bullet d, \$]$. Riassumendo:

$$goto(I_0, C) = closure(\{[S \to C \bullet C, \$]\}) = \{[S \to C \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_2$$

Calcolo della *closure*(I): un esempio

$goto(I_0, c) = closure(\{[C \rightarrow c \bullet C, c], C \rightarrow c \bullet C, d]\})$

- inanzitutto, aggiungiamo gli items $[C \rightarrow c \bullet C, c]$ e $[C \rightarrow c \bullet C, d]$
- l'item $[C \rightarrow c \bullet C, c]$ causa l'inserimento di kernel items (uno o più) per ciascuna produzione per C. Il simbolo di lookahead deve appartenere alla FIRST(βa) con $\beta = \varepsilon$ e $a = c$ (quindi FIRST($\beta\, a$) $= \{c\}$).
- dobbiamo aggiunge gli items $[C \rightarrow \bullet cC, c], [C \rightarrow \bullet d, c]$.
- in maniera simile, dall'item $[C \rightarrow c \bullet C, d]$ (qui $\beta = \varepsilon$, $a = d$ e quindi FIRST($\beta\, a$) $= \{d\}$) aggiungiamo altri due items: $[C \rightarrow \bullet cC, d]$ e $[C \rightarrow \bullet d, d]$.

Calcolo della *closure*(I): un esempio

Ricapitolando:

$$goto(I_0, c) = closure(\{[C \rightarrow c \bullet C, c], C \rightarrow c \bullet C, d]\}) =$$
$$\{$$
$$[C \rightarrow c \bullet C, c], C \rightarrow c \bullet C, d],$$
$$[C \rightarrow \bullet cC, c], [C \rightarrow \bullet d, c]$$
$$[C \rightarrow \bullet cC, d], [C \rightarrow \bullet d, d]$$
$$\}$$

o in forma abbreviata

$$goto(I_0, c) = closure(\{[C \rightarrow c \bullet C, c/d]\}) =$$
$$\{$$
$$[C \rightarrow c \bullet C, c/d],$$
$$[C \rightarrow \bullet cC, c/d], [C \rightarrow \bullet d, c/d]$$
$$\} = I_3$$

Calcolo della *closure*(I): un esempio

In maniera del tutto simile:

$$goto(I_0, d) = closure(\{[C \rightarrow d\bullet, c/d]\}) =$$
$$\{$$
$$\quad [C \rightarrow d\bullet, c/d]$$
$$\} = I_4$$

Questo conclude il calcolo della *goto* "uscenti" da I_0; $I_1 = \{[S' \rightarrow S\bullet, \$]\}$

Possiamo passare a I_2

$$goto(I_2, C) = closure(\{[S \rightarrow CC\bullet, \$]\}) = \{[S \rightarrow CC\bullet, \$]\} = I_5$$

Calcolo della *closure*(I): un esempio

$$goto(I_2, c) = closure(\{[C \rightarrow c \bullet C, \$]\}) =$$
$$\{$$
$$\quad [C \rightarrow c \bullet C, \$],$$
$$\quad [C \rightarrow \bullet cC, \$],$$
$$\quad [C \rightarrow \bullet d, \$]$$
$$\} = I_6$$

$$goto(I_2, d) = closure(\{[C \rightarrow d\bullet, \$]\}) =$$
$$\{$$
$$\quad [C \rightarrow d\bullet, \$]$$
$$\} = I_7$$

Calcolo della *closure*(I): un esempio

$$goto(I_3, C) = closure(\{[C \rightarrow cC\bullet, c], C \rightarrow cC\bullet, d]\}) =$$
$$\{$$
$$\quad [C \rightarrow cC\bullet, c], C \rightarrow cC\bullet, d]$$
$$\} = I_8$$

$$goto(I_3, c) = closure(\{[C \rightarrow c \bullet C, c/d]\}) = I_3$$

$$goto(I_3, d) = closure(\{[C \rightarrow d\bullet, c/d]\}) = I_4$$

$$goto(I_6, C) = closure(\{[C \rightarrow cC\bullet, \$]\}) = \{[C \rightarrow cC\bullet, \$]\} = I_9$$
$$goto(I_6, c) = closure(\{[C \rightarrow c \bullet C, \$]\}) = I_6$$
$$goto(I_6, d) = closure(\{[C \rightarrow d\bullet, \$]\}) = I_7$$

Alcune considerazioni

- Consideriamo gli insiemi di item LR(1) $I_4 = \{[C \rightarrow d\bullet, c/d]\}$ ed $I_7 = \{[C \rightarrow d\bullet, \$]\}$
- Questi insiemi sono molto particolari perchè differiscono solo per la seconda componente
- Questo fenomeno è tipico per gli insiemi di item LR(1)
- Un insieme della collezione canonica LR(0), ad esempio $\{C \rightarrow d\bullet\}$, può corrispondere in generale a più insiemi di item LR(1), in questo caso I_4 ed I_7
- È come se gli stati del parser SLR venissero sdoppiati

Costruzione di tabelle LR canoniche

- Il procedimento è analogo a quello visto per la costruzione di tabelle SLR, anzi, è più semplice
- Per le riduzioni dobbiamo guardare il simbolo di lookahead dell'item e non tutti i simboli in FOLLOW(A) dove A è la parte sinistra della produzione con cui ridurre

Algoritmo

1. Costruisci la collezione canonica $C = \{I_0, I_1, \ldots, I_n\}$ per G'
2. Lo stato s_j del parser corrisponde all'insieme di items I_j
3. La parte *goto* della tabella, per ogni stato s_j e per ogni non terminale A, è costruita dalla funzione *goto* come segue:

   se $goto(I_j, A) = I_k$ (funzione) allora $goto[s_j, A] = s_k$ (tabella)
4. La parte *action* della tabella è costruita come segue:
   - se $[A \to \alpha \bullet a\beta, b] \in I_j$ e $goto(I_j, a) = I_k$ allora $action[s_j, a] =$ shift s_k (qui a è un terminale)
   - se $[A \to \alpha\bullet, a] \in I_j$, allora poni $action[s_j, a] =$ reduce $A \to \alpha$
   - se $[S' \to S\bullet, \$] \in I_j$, allora poni $action[s_j, \$] =$ accept
5. Ogni entrata indefinita corrisponde ad un errore
6. Lo stato iniziale corrisponde all'insieme che contiene $[S' \to \bullet S, \$]$

Costruzione di tabelle LR canoniche

- Come nel caso SLR, la parte goto della tabella LR è definita dalla funzione *goto*
- Le entrate "shift" si costruiscono esattamente allo stesso modo
- Come detto, in caso di riduzione, guardiamo il simbolo di lookahead dell'item
- Lo stato iniziale corrisponde all'insieme che contiene $[S' \rightarrow \bullet S, \$]$
- Tutte le entrate vuote corrispondono a "error"
- La tabella così costruita si chiama **tabella di parsing LR(1) canonica**
- Se nella tabella non ci sono entrate multidefinite allora la grammatica è una grammatica LR(1)
- Un parser LR che utilizza questa tabella si chiama parser LR(1) canonico

Costruzione della tabella: un esempio

Numeriamo le produzioni della grammatica e costruiamo la tabella

$$(0)\ S' \to S \qquad (1)\ S \to CC \qquad (2)\ C \to cC \qquad (3)\ C \to d$$

$goto(I_0, S) = \{[S' \to S\bullet, \$]\} = I_1$
$goto(I_0, C) = \{[S \to C \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_2$
$goto(I_0, c) = \{[C \to c \bullet C, c/d], [C \to \bullet cC, c/d], [C \to \bullet d, c/d]\} = I_3$
$goto(I_0, d) = \{[C \to d\bullet, c/d]\} = I_4$

$goto(I_2, C) = \{[S \to CC\bullet, \$]\} = I_5$
$goto(I_2, c) = \{[C \to c \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_6$
$goto(I_2, d) = \{[C \to d\bullet, \$]\} = I_7$

$goto(I_3, C) = \{[C \to cC\bullet, c/d]\} = I_8$
$goto(I_3, c) = I_3, \quad goto(I_3, d) = I_4$

$goto(I_6, C) = \{[C \to cC\bullet, \$]\} = I_9$
$goto(I_6, c) = I_6, \quad goto(I_6, d) = I_7$

Costruzione della tabella: un esempio

Numeriamo le produzioni della grammatica e costruiamo la tabella

$$(0)\ S' \to S \qquad (1)\ S \to CC \qquad (2)\ C \to cC \qquad (3)\ C \to d$$

$goto(I_0, S) = \{[S' \to S\bullet, \$]\} = I_1$
$goto(I_0, C) = \{[S \to C \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_2$
$goto(I_0, c) = \{[C \to c \bullet C, c/d], [C \to \bullet cC, c/d], [C \to \bullet d, c/d]\} = I_3$
$goto(I_0, d) = \{[C \to d\bullet, c/d]\} = I_4$

$goto(I_2, C) = \{[S \to CC\bullet, \$]\} = I_5$
$goto(I_2, c) = \{[C \to c \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_6$
$goto(I_2, d) = \{[C \to d\bullet, \$]\} = I_7$

$goto(I_3, C) = \{[C \to cC\bullet, c/d]\} = I_8$
$goto(I_3, c) = I_3, \quad goto(I_3, d) = I_4$

$goto(I_6, C) = \{[C \to cC\bullet, \$]\} = I_9$
$goto(I_6, c) = I_6, \quad goto(I_6, d) = I_7$

Costruzione della tabella: un esempio

Numeriamo le produzioni della grammatica e costruiamo la tabella

$$(0)\ S' \to S \qquad (1)\ S \to CC \qquad (2)\ C \to cC \qquad (3)\ C \to d$$

$goto(I_0, S) = \{[S' \to S\bullet, \$]\} = I_1$
$goto(I_0, C) = \{[S \to C \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_2$
$goto(I_0, c) = \{[C \to c \bullet C, c/d], [C \to \bullet cC, c/d], [C \to \bullet d, c/d]\} = I_3$
$goto(I_0, d) = \{[C \to d\bullet, c/d]\} = I_4$

$goto(I_2, C) = \{[S \to CC\bullet, \$]\} = I_5$
$goto(I_2, c) = \{[C \to c \bullet C, \$], [C \to \bullet cC, \$], [C \to \bullet d, \$]\} = I_6$
$goto(I_2, d) = \{[C \to d\bullet, \$]\} = I_7$

$goto(I_3, C) = \{[C \to cC\bullet, c/d]\} = I_8$
$goto(I_3, c) = I_3, \quad goto(I_3, d) = I_4$

$goto(I_6, C) = \{[C \to cC\bullet, \$]\} = I_9$
$goto(I_6, c) = I_6, \quad goto(I_6, d) = I_7$

La tabella

	c	d	\$	S	C
s_0	$S3$	$S4$		1	2
s_1			ACC		
s_2	S6	S7			5
s_3	S3	S4			8
s_4	R3	R3			
s_5			R1		
s_6	S6	S7			9
s_7			R3		
s_8	R2	R2			
s_9			R2		

Parsing della stringa *ccdcd*

	c	d	\$	S	C
s_0	$S3$	$S4$		1	2
s_1			ACC		
s_2	S6	S7			5
s_3	S3	S4			8
s_4	R3	R3			
s_5			R1		
s_6	S6	S7			9
s_7			R3		
s_8	R2	R2			
s_9			R2		

Stack	Input	Azione
s_0	$ccdcd\$$	shift S3
$s_0\, c\, s_3$	$cdcd\$$	shift S3
$s_0\, c\, s_3\, c\, s_3$	$dcd\$$	shift S4
$s_0\, c\, s_3\, c\, s_3\, d s_4$	$cd\$$	red. $C \rightarrow d$
$s_0\, c\, s_3\, c\, s_3\, C s_8$	$cd\$$	red. $C \rightarrow cC$
$s_0\, c\, s_3\, C s_8$	$cd\$$	red. $C \rightarrow cC$
$s_0\, C s_2$	$cd\$$	shift S6
$s_0\, C s_2\, c\, s_6$	$d\$$	shift S7
$s_0\, C s_2\, c\, s_6\, d\, s_7$	$\$$	red. $C \rightarrow d$
$s_0\, C s_2\, c\, s_6\, C\, s_9$	$\$$	red. $C \rightarrow cC$
$s_0\, C s_2\, C s_5$	$\$$	red. $S \rightarrow CC$
$s_0 S s_1$	$\$$	acc