Lecture 23 - Dec. 6

Syntactic Analysis

Algorithms: BuildCC, BuildTables
Conflicts: shift-reduce vs. reduce-reduce

Announcements

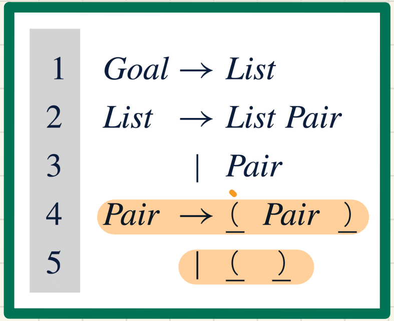
- **Project** final submission tonight!
- **Review session** at 1pm on Thursday, December 8

CC Construction: goto

→ cc₀ $\xrightarrow{List}$?

Calculate goto(cc₀, List)

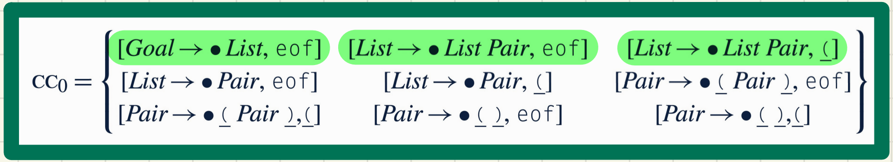
i.e., “next subset state” from cc₀ taking ~~(~~ List



1	$Goal \rightarrow List$
2	$List \rightarrow List\ Pair$
3	$\quad\mid Pair$
4	$Pair \rightarrow (\ Pair\ )$
5	$\quad\mid (\ )$

closure({ [Goal → List • eof],
[List → List • Pair, eof],
[List → List • Pair, ( ] })

?



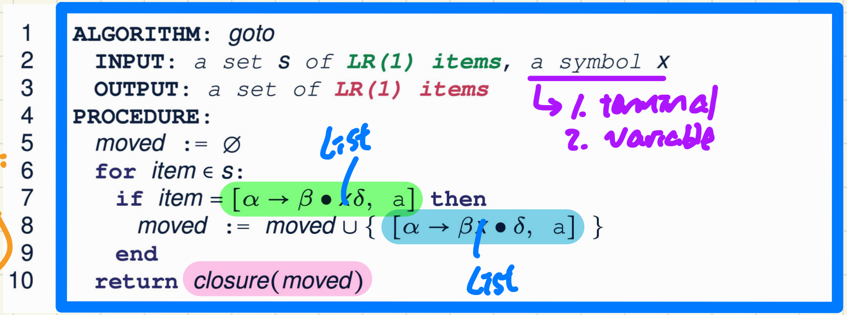
$$CC_0 = \left\{ \begin{array}{ccc} [Goal \rightarrow \bullet\ List, \texttt{eof}] & [List \rightarrow \bullet\ List\ Pair, \texttt{eof}] & [List \rightarrow \bullet\ List\ Pair, (] \\ [List \rightarrow \bullet\ Pair, \texttt{eof}] & [List \rightarrow \bullet\ Pair, (] & [Pair \rightarrow \bullet\ (\ Pair\ ), \texttt{eof}] \\ [Pair \rightarrow \bullet\ (\ Pair\ ), (] & [Pair \rightarrow \bullet\ (\ ), \texttt{eof}] & [Pair \rightarrow \bullet\ (\ ), (] \end{array} \right\}$$

Dimension 1: Two alt. for Pair

Pair → (Pair)

Pair → ( )

Dimension 2: FIRST(δa)

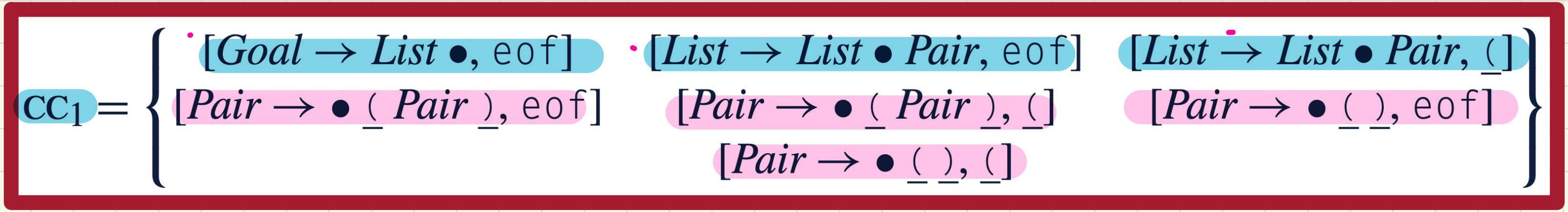


```
ALGORITHM: goto
  INPUT: a set S of LR(1) items, a symbol x
  OUTPUT: a set of LR(1) items
PROCEDURE:
  moved := ∅
  for item ∈ S:
    if item = $[\alpha \rightarrow \beta \bullet x\delta,\ a]$ then
      moved := moved ∪ { $[\alpha \rightarrow \beta x \bullet \delta,\ a]$ }
    end
  return closure(moved)
```

↳ 1. terminal
2. variable

List

List



$$CC_1 = \left\{ \begin{array}{ccc} [Goal \rightarrow List\ \bullet, \texttt{eof}] & [List \rightarrow List\ \bullet\ Pair, \texttt{eof}] & [List \rightarrow List\ \bullet\ Pair, (] \\ [Pair \rightarrow \bullet\ (\ Pair\ ), \texttt{eof}] & [Pair \rightarrow \bullet\ (\ Pair\ ), (] & [Pair \rightarrow \bullet\ (\ ), \texttt{eof}] \\ & [Pair \rightarrow \bullet\ (\ ), (] & \end{array} \right\}$$

CC and δ Construction: Algorithm and Exercise

```
ALGORITHM: BuildCC
  INPUT: a grammar $G = (V, \Sigma, R, S)$, goal production $S \rightarrow S'$
  OUTPUT:
    (1) a set $CC = \{cc_0, cc_1, \ldots, cc_n\}$ where $cc_i \subseteq G'$s $LR(1)$ items
    (2) a transition function
PROCEDURE:
  $cc_0$ := closure({[$S \rightarrow \bullet S$, eof]})
  $CC$ := {$cc_0$}
  processed := {$cc_0$}
  lastCC := $\varnothing$
  while (lastCC $\neq$ CC):
    lastCC := CC
    for $cc_i$ s.t. $cc_i \in CC \wedge cc_i \notin$ processed:
      processed := processed $\cup$ {$cc_i$}
      for x s.t. [$\cdots \rightarrow \cdots \bullet$ x $\ldots$] $\in cc_i$
        temp := goto($cc_i$, x)
        if temp $\notin$ CC then
          CC := CC $\cup$ {temp}
        end
        $\delta$ := $\delta \cup$ ($cc_i$, x, temp)
```

start var.

x '

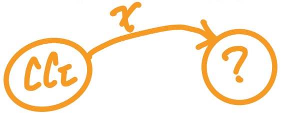
make a transition from CC_i via recognizing x

ready to recognize a terminal or variable

src state

tar state

transition



1	Goal	$\rightarrow$	List
2	List	$\rightarrow$	List Pair
3		|	Pair
4	Pair	$\rightarrow$	(Pair)
5		|	( )

Ex1. Calculate CC (i.e., all reachable subset states).

Ex2. Calculate δ (i.e., relating members of CC by terminals and non-terminals).

CC and δ Construction: Output 1

$$CC_0 = \left\{ \begin{array}{ccc} [Goal \rightarrow \bullet List, \texttt{eof}] & [List \rightarrow \bullet List\ Pair, \texttt{eof}] & [List \rightarrow \bullet List\ Pair, \underline{(}] \\ [List \rightarrow \bullet Pair, \texttt{eof}] & [List \rightarrow \bullet Pair, \underline{(}] & [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \texttt{eof}] \\ [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \underline{(}] & [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \texttt{eof}] & [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \underline{(}] \end{array} \right\}$$

List

$$CC_1 = \left\{ \begin{array}{ccc} [Goal \rightarrow List \bullet, \texttt{eof}] & [List \rightarrow List \bullet Pair, \texttt{eof}] & [List \rightarrow List \bullet Pair, \underline{(}] \\ [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \texttt{eof}] & [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \underline{(}] & [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \texttt{eof}] \\ & [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \underline{(}] & \end{array} \right\}$$

$$CC_2 = \left\{ [List \rightarrow Pair \bullet, \texttt{eof}] \quad [List \rightarrow Pair \bullet, \underline{(}] \right\}$$

(

$$CC_3 = \left\{ \begin{array}{ccc} [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \underline{)}] & [Pair \rightarrow \underline{(} \bullet Pair\ \underline{)}, \texttt{eof}] & [Pair \rightarrow \underline{(} \bullet Pair\ \underline{)}, \underline{(}] \\ [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \underline{)}] & [Pair \rightarrow \underline{(} \bullet \underline{)}, \texttt{eof}] & [Pair \rightarrow \underline{(} \bullet \underline{)}, \underline{(}] \end{array} \right\}$$

$$CC_4 = \left\{ [List \rightarrow List\ Pair \bullet, \texttt{eof}] \quad [List \rightarrow List\ Pair \bullet, \underline{(}] \right\}$$

$$CC_5 = \left\{ [Pair \rightarrow \underline{(}\ Pair \bullet \underline{)}, \texttt{eof}] \quad [Pair \rightarrow \underline{(}\ Pair \bullet \underline{)}, \underline{(}] \right\}$$

$$CC_6 = \left\{ \begin{array}{cc} [Pair \rightarrow \bullet \underline{(}\ Pair\ \underline{)}, \underline{)}] & [Pair \rightarrow \underline{(} \bullet Pair\ \underline{)}, \underline{)}] \\ [Pair \rightarrow \bullet \underline{(}\ \underline{)}, \underline{)}] & [Pair \rightarrow \underline{(} \bullet \underline{)}, \underline{)}] \end{array} \right\}$$

$$CC_7 = \left\{ [Pair \rightarrow \underline{(}\ \underline{)} \bullet, \texttt{eof}] \quad [Pair \rightarrow \underline{(}\ \underline{)} \bullet, \underline{(}] \right\}$$

$$CC_8 = \left\{ [Pair \rightarrow \underline{(}\ Pair\ \underline{)} \bullet, \texttt{eof}] \quad [Pair \rightarrow \underline{(}\ Pair\ \underline{)} \bullet, \underline{(}] \right\}$$

$$CC_9 = \left\{ [Pair \rightarrow \underline{(}\ Pair \bullet \underline{)}, \underline{)}] \right\}$$

$$CC_{10} = \left\{ [Pair \rightarrow \underline{(}\ \underline{)} \bullet, \underline{)}] \right\}$$

$$CC_{11} = \left\{ [Pair \rightarrow \underline{(}\ Pair\ \underline{)} \bullet, \underline{)}] \right\}$$

CC and δ Construction: Output 2

Transition Function

Iteration	Item	*Goal*	*List*	*Pair*	(	)	eof
0	cc_0	∅	cc_1	cc_2	cc_3	∅	∅
1	cc_1	∅	∅	cc_4	cc_3	∅	∅
	cc_2	∅	∅	∅	∅	∅	∅
	cc_3	∅	∅	cc_5	cc_6	cc_7	∅
2	cc_4	∅	∅	∅	∅	∅	∅
	cc_5	∅	∅	∅	∅	cc_8	∅
	cc_6	∅	∅	cc_9	cc_6	cc_{10}	∅
	cc_7	∅	∅	∅	∅	∅	∅
3	cc_8	∅	∅	∅	∅	∅	∅
	cc_9	∅	∅	∅	∅	cc_{11}	∅
	cc_{10}	∅	∅	∅	∅	∅	∅
4	cc_{11}	∅	∅	∅	∅	∅	∅

DFA of the LR(1) Parser

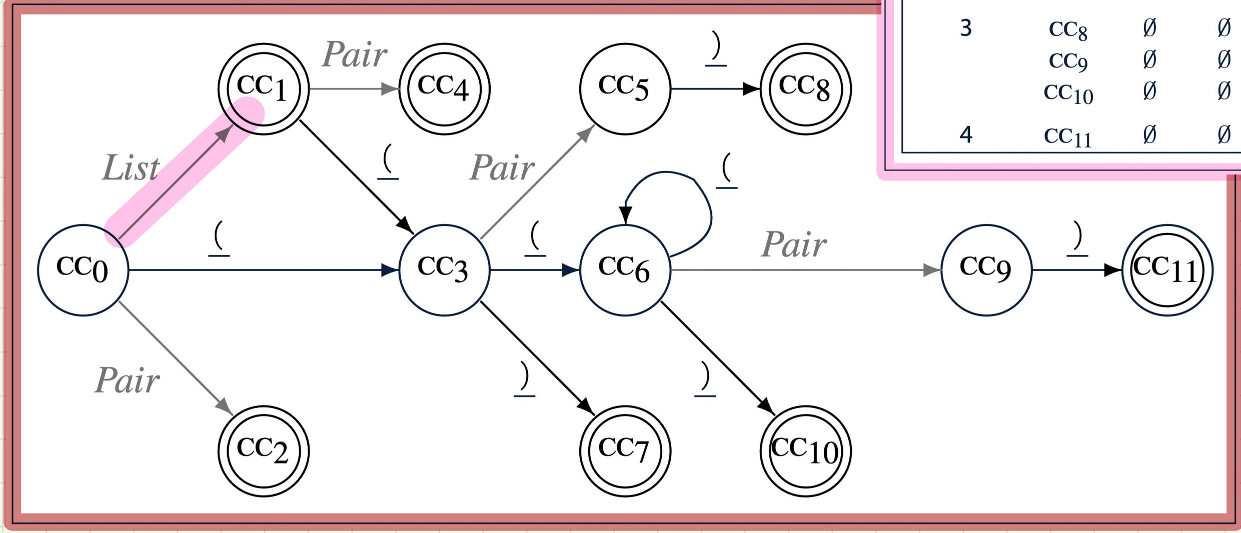


Table Construction: Algorithm

```
ALGORITHM: BuildActionGotoTables
  INPUT:
    (1) a grammar $G = (V, \Sigma, R, S)$
    (2) goal production $S \to S'$
    (3) a canonical collection $CC = \{cc_0, cc_1, \ldots, cc_n\}$
    (4) a transition function $\delta : CC \times \Sigma \to CC$
  OUTPUT: Action Table & Goto Table
PROCEDURE:
  for $cc_i \in CC$:
    for $item \in cc_i$:
      if $item = [A \to \beta \bullet x\gamma,\ a] \land \delta(cc_i, x) = cc_j$ then
        Action[i, x] := shift j
      elseif $item = [A \to \beta \bullet,\ a]$ then
        Action[i, a] := reduce $A \to \beta$
      elseif $item = [S \to S' \bullet,\ eof]$ then
        Action[i, eof] := accept
      end
    for $v \in V$:
      if $\delta(cc_i, v) = cc_j$ then
        Goto[i, v] := j
      end
```

produced by BuildCC

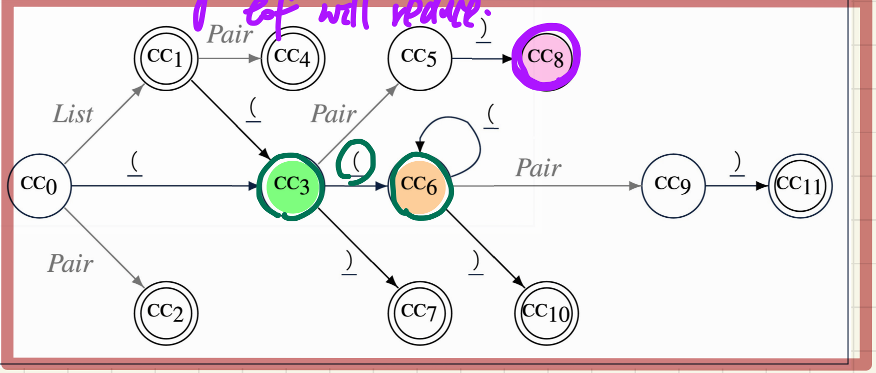
$\delta(CC_3, ( ) = CC_6$

i j 3 6

CC8 is already an accepting state, meaning reading eof will reduce.

fill in Goto table

State	Action Table eof	(	)	Goto Table List	Pair
0		s 3		1	2
1	acc	s 3			4
2	r 3	r 3			
3		s 6	s 7		5
4	r 2	r 2			
5			s 8		
6		s 6	s 10		9
7	r 5	r 5			
8	r 4	r 4			
9			s 11		
10			r 5		
11			r 4		



$$cc_8 = \{[Pair \to (\ Pair \ ) \bullet, eof] \quad [Pair \to (\ Pair \ ) \bullet, (]\}$$

Bottom-Up Parsing: Discovering Ambiguities

$$CC_{13} = \left\{ \begin{array}{l} [Stmt \rightarrow \texttt{if expr then } Stmt \bullet, \{\texttt{eof}, \texttt{else}\}], \\ [Stmt \rightarrow \texttt{if expr then } Stmt \bullet \texttt{else } Stmt, \{\texttt{eof}, \texttt{else}\}] \end{array} \right\}$$

certain state of paser

by reading eof or else, reduce to Stmt

by reading else, we shift to

What if the current word to match is else?

γδ already recognized

shift or reduce to Stmt

↳ shift-reduce conflict

↳ in practice, shift will be done.

$$CC_i = \left\{ \begin{array}{l} [A \rightarrow \gamma\delta \bullet, \texttt{a}], \\ [B \rightarrow \gamma\delta \bullet, \texttt{a}] \end{array} \right\}$$

by readin a,

What if the current word to match is a?

some reduction

↳ reduce-reduce conflict → must fix the grammar.

Exam.

1. no multiple choice questions
2. no data sheets (algorithms included)
3. format similar to quizzes
4. cumulative.