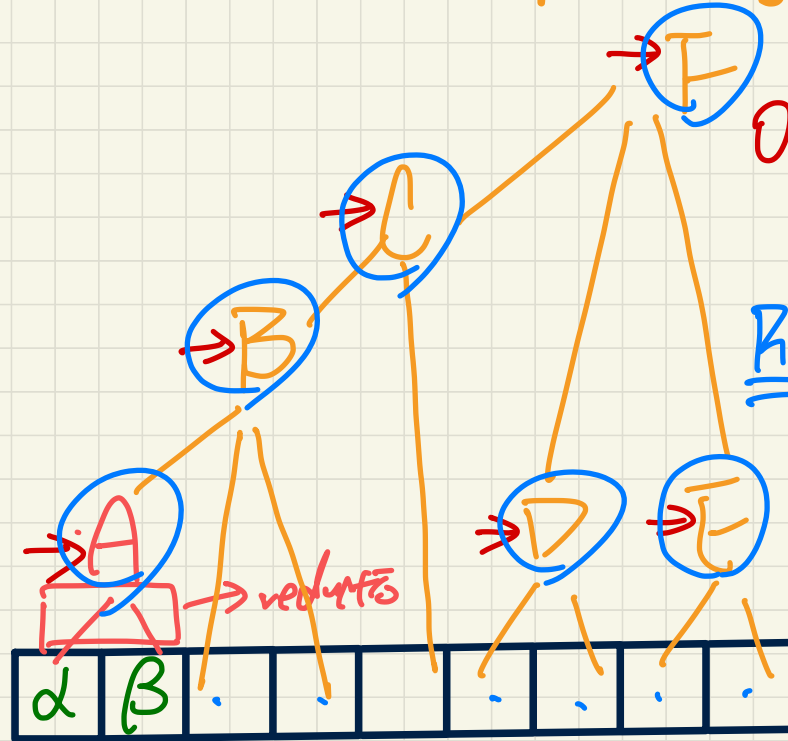


Discovering Derivations: Bottom-Up Parsing

production rule

$$A \rightarrow \alpha \beta$$

scanner



Order of reductions:
A B C D E F.

RMD:
F E D C B A

Bottom-Up Parsing: Algorithm

ALGORITHM: *BU_{Parse}*

INPUT: CFG $G = (V, \Sigma, R, S)$, **Action** & **Goto** Tables

OUTPUT: Report Parse Success or Syntax Error

PROCEDURE:

initialize an empty stack *trace*

trace.push(0) /* start state */

word := NextWord()

while (true)

state := *trace*.top()

act := **Action**[state, word]

if act = "accept" then

succeed()

elseif act = "reduce based on $A \rightarrow \beta$ " then

trace.pop() $2 \times |\beta|$ times /* word + state */

state := *trace*.top()

trace.push(A)

next := **Goto**[state, A]

trace.push(next)

elseif act = "shift to s" then

trace.push(word)

trace.push(i)

word := NextWord()

else

fail()

1 Goal $\rightarrow$ List

2 List $\rightarrow$ List Pair

3 | Pair

4 Pair $\rightarrow$ (Pair)

5 | ()

State	Action Table			Goto Table	
	eof	(	)	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		

Bottom-Up Parsing: Discovering Rightmost Derivations (1)

ALGORITHM: *BUParse*

INPUT: *CFG* $G = (V, \Sigma, R, S)$, **Action** & **Goto** Tables

OUTPUT: *Report Parse Success* or *Syntax Error*

PROCEDURE :

```
initialize an empty stack trace
```

```
trace.push(0) /* start state */
```

```
word := NextWord()
```

```
while (true)
```

```
→ state := trace.top() "C"
```

$\text{act} := \text{Action}[\text{state}, \text{word}]$

```
if act == 'accept' then
```

→ `succeed()`

```
elseif  $act = \text{"reduce based on } A \rightarrow \beta \text{"}$  then
```

- `trace.pop()` $2 \times |\beta|$ times /* word +

```
state := trace.top()
```

```
trace.push(A)
```

```
next := Goto[state, A]
```

```

next = None
trace.push(next)

```

```
elseif act == ``shift to s:`` then
```

```

    trace push(word

```

trace.push(i) ✓

```
trace.push(1) ✓ ✓
word := NextWord()
```

also

```
else
    fail()
```

1 $Goal \rightarrow List$

2 *List* $\rightarrow$ *List Pair*

3 | *Pair*

4 $Pair \rightarrow (Pair)$

5 | ()

State	Action Table			Goto Table	
	eof	(	)	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		

Parse: () ✓

word: "x" "x" of
state: ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ ~~12~~ ~~13~~ ~~14~~ ~~15~~ ~~16~~ ~~17~~ ~~18~~ ~~19~~ ~~20~~ ~~21~~ ~~22~~ ~~23~~ ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ ~~29~~ ~~30~~ ~~31~~ ~~32~~ ~~33~~ ~~34~~ ~~35~~ ~~36~~ ~~37~~ ~~38~~ ~~39~~ ~~40~~ ~~41~~ ~~42~~ ~~43~~ ~~44~~ ~~45~~ ~~46~~ ~~47~~ ~~48~~ ~~49~~ ~~50~~ ~~51~~ ~~52~~ ~~53~~ ~~54~~ ~~55~~ ~~56~~ ~~57~~ ~~58~~ ~~59~~ ~~60~~ ~~61~~ ~~62~~ ~~63~~ ~~64~~ ~~65~~ ~~66~~ ~~67~~ ~~68~~ ~~69~~ ~~70~~ ~~71~~ ~~72~~ ~~73~~ ~~74~~ ~~75~~ ~~76~~ ~~77~~ ~~78~~ ~~79~~ ~~80~~ ~~81~~ ~~82~~ ~~83~~ ~~84~~ ~~85~~ ~~86~~ ~~87~~ ~~88~~ ~~89~~ ~~90~~ ~~91~~ ~~92~~ ~~93~~ ~~94~~ ~~95~~ ~~96~~ ~~97~~ ~~98~~ ~~99~~ ~~100~~ ~~101~~ ~~102~~ ~~103~~ ~~104~~ ~~105~~ ~~106~~ ~~107~~ ~~108~~ ~~109~~ ~~110~~ ~~111~~ ~~112~~ ~~113~~ ~~114~~ ~~115~~ ~~116~~ ~~117~~ ~~118~~ ~~119~~ ~~120~~ ~~121~~ ~~122~~ ~~123~~ ~~124~~ ~~125~~ ~~126~~ ~~127~~ ~~128~~ ~~129~~ ~~130~~ ~~131~~ ~~132~~ ~~133~~ ~~134~~ ~~135~~ ~~136~~ ~~137~~ ~~138~~ ~~139~~ ~~140~~ ~~141~~ ~~142~~ ~~143~~ ~~144~~ ~~145~~ ~~146~~ ~~147~~ ~~148~~ ~~149~~ ~~150~~ ~~151~~ ~~152~~ ~~153~~ ~~154~~ ~~155~~ ~~156~~ ~~157~~ ~~158~~ ~~159~~ ~~160~~ ~~161~~ ~~162~~ ~~163~~ ~~164~~ ~~165~~ ~~166~~ ~~167~~ ~~168~~ ~~169~~ ~~170~~ ~~171~~ ~~172~~ ~~173~~ ~~174~~ ~~175~~ ~~176~~ ~~177~~ ~~178~~ ~~179~~ ~~180~~ ~~181~~ ~~182~~ ~~183~~ ~~184~~ ~~185~~ ~~186~~ ~~187~~ ~~188~~ ~~189~~ ~~190~~ ~~191~~ ~~192~~ ~~193~~ ~~194~~ ~~195~~ ~~196~~ ~~197~~ ~~198~~ ~~199~~ ~~200~~ ~~201~~ ~~202~~ ~~203~~ ~~204~~ ~~205~~ ~~206~~ ~~207~~ ~~208~~ ~~209~~ ~~210~~ ~~211~~ ~~212~~ ~~213~~ ~~214~~ ~~215~~ ~~216~~ ~~217~~ ~~218~~ ~~219~~ ~~220~~ ~~221~~ ~~222~~ ~~223~~ ~~224~~ ~~225~~ ~~226~~ ~~227~~ ~~228~~ ~~229~~ ~~230~~ ~~231~~ ~~232~~ ~~233~~ ~~234~~ ~~235~~ ~~236~~ ~~237~~ ~~238~~ ~~239~~ ~~240~~ ~~241~~ ~~242~~ ~~243~~ ~~244~~ ~~245~~ ~~246~~ ~~247~~ ~~248~~ ~~249~~ ~~250~~ ~~251~~ ~~252~~ ~~253~~ ~~254~~ ~~255~~ ~~256~~ ~~257~~ ~~258~~ ~~259~~ ~~260~~ ~~261~~ ~~262~~ ~~263~~ ~~264~~ ~~265~~ ~~266~~ ~~267~~ ~~268~~ ~~269~~ ~~270~~ ~~271~~ ~~272~~ ~~273~~ ~~274~~ ~~275~~ ~~276~~ ~~277~~ ~~278~~ ~~279~~ ~~280~~ ~~281~~ ~~282~~ ~~283~~ ~~284~~ ~~285~~ ~~286~~ ~~287~~ ~~288~~ ~~289~~ ~~290~~ ~~291~~ ~~292~~ ~~293~~ ~~294~~ ~~295~~ ~~296~~ ~~297~~ ~~298~~ ~~299~~ ~~300~~ ~~301~~ ~~302~~ ~~303~~ ~~304~~ ~~305~~ ~~306~~ ~~307~~ ~~308~~ ~~309~~ ~~310~~ ~~311~~ ~~312~~ ~~313~~ ~~314~~ ~~315~~ ~~316~~ ~~317~~ ~~318~~ ~~319~~ ~~320~~ ~~321~~ ~~322~~ ~~323~~ ~~324~~ ~~325~~ ~~326~~ ~~327~~ ~~328~~ ~~329~~ ~~330~~ ~~331~~ ~~332~~ ~~333~~ ~~334~~ ~~335~~ ~~336~~ ~~337~~ ~~338~~ ~~339~~ ~~340~~ ~~341~~ ~~342~~ ~~343~~ ~~344~~ ~~345~~ ~~346~~ ~~347~~ ~~348~~ ~~349~~ ~~350~~ ~~351~~ ~~352~~ ~~353~~ ~~354~~ ~~355~~ ~~356~~ ~~357~~ ~~358~~ ~~359~~ ~~360~~ ~~361~~ ~~362~~ ~~363~~ ~~364~~ ~~365~~ ~~366~~ ~~367~~ ~~368~~ ~~369~~ ~~370~~ ~~371~~ ~~372~~ ~~373~~ ~~374~~ ~~375~~ ~~376~~ ~~377~~ ~~378~~ ~~379~~ ~~380~~ ~~381~~ ~~382~~ ~~383~~ ~~384~~ ~~385~~ ~~386~~ ~~387~~ ~~388~~ ~~389~~ ~~390~~ ~~391~~ ~~392~~ ~~393~~ ~~394~~ ~~395~~ ~~396~~ ~~397~~ ~~398~~ ~~399~~ ~~400~~ ~~401~~ ~~402~~ ~~403~~ ~~404~~ ~~405~~ ~~406~~ ~~407~~ ~~408~~ ~~409~~ ~~410~~ ~~411~~ ~~412~~ ~~413~~ ~~414~~ ~~415~~ ~~416~~ ~~417~~ ~~418~~ ~~419~~ ~~420~~ ~~421~~ ~~422~~ ~~423~~ ~~424~~ ~~425~~ ~~426~~ ~~427~~ ~~428~~ ~~429~~ ~~430~~ ~~431~~ ~~432~~ ~~433~~ ~~434~~ ~~435~~ ~~436~~ ~~437~~ ~~438~~ ~~439~~ ~~440~~ ~~441~~ ~~442~~ ~~443~~ ~~444~~ ~~445~~ ~~446~~ ~~447~~ ~~448~~ ~~449~~ ~~450~~ ~~451~~ ~~452~~ ~~453~~ ~~454~~ ~~455~~ ~~456~~ ~~457~~ ~~458~~ ~~459~~ ~~460~~ ~~461~~ ~~462~~ ~~463~~ ~~464~~ ~~465~~

handle

1. List

twice

Bottom-Up Parsing: Discovering Rightmost Derivations (2)

ALGORITHM: *BUParse*

INPUT: CFG $G = (V, \Sigma, R, S)$, Action & Goto Tables

OUTPUT: Report Parse Success or Syntax Error

PROCEDURE:

initialize an empty stack *trace*

trace.push(0) /* start state */

word := *NextWord*()

while (**true**)

state := *trace.top*()

act := **Action**[*state*, *word*]

if *act* = ``accept'' **then**

succeed()

elseif *act* = ``reduce based on $A \rightarrow \beta$ '' **then**

trace.pop() $2 \times |\beta|$ times /* word +

state := *trace.top*()

trace.push(*A*)

next := **Goto**[*state*, *A*]

trace.push(*next*)

elseif *act* = ``shift to s_i '' **then**

trace.push(*word*)

trace.push(*i*)

word := *NextWord*()

else

fail()

Parse: (()) ()

```
1  Goal → List
2  List  → List Pair
3         | Pair
4  Pair  → ( Pair )
5         | (   )
```

State	Action Table			Goto Table	
	eof	(	)	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		

Bottom-Up Parsing: Discovering Rightmost Derivations (3)

ALGORITHM: *BUParse*

INPUT: CFG $G = (V, \Sigma, R, S)$, *Action* & *Goto* Tables

OUTPUT: *Report Parse Success* or *Syntax Error*

PROCEDURE:

initialize an empty stack *trace*

trace.push(0) /* start state */

word := *NextWord*()

while (**true**)

state := *trace.top*()

act := *Action*[*state*, *word*]

if *act* = ``accept'' **then**

succeed()

elseif *act* = ``reduce based on $A \rightarrow \beta$ '' **then**

trace.pop() $2 \times |\beta|$ times /* word +

state := *trace.top*()

trace.push(*A*)

next := *Goto*[*state*, *A*]

trace.push(*next*)

elseif *act* = ``shift to s_i '' **then**

trace.push(*word*)

trace.push(*i*)

word := *NextWord*()

else

fail()

Parse: ())

```
1  Goal → List
2  List → List Pair
3      | Pair
4  Pair → ( Pair )
5      | ( )
```

State	Action Table			Goto Table	
	eof	(	)	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		

Bottom-Up Parsing: Handles

& Goto Tables

ntax Error

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

 β'' then

	Action Table			Goto Table	
State	eof	(	)	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 10		9
7	r 5	s 5			
8	r 4	r 4			
9			s 11		
10			r 5		
11			r 4		

Parse: () ✓

word: "X" "X" of:
state: X X X X X X X

handle

1. List

twice

A **handle** denotes a parser's state that's ready for **reduction**.

Iteration	State	word	Stack	Handle	Action
<i>initial</i>	—	(	\$ 0	<i>none</i>	—
1	0	(	\$ 0	<i>none</i>	shift 3
2	3	)	\$ 0 (3	<i>none</i>	shift 7
3	7	eof	\$ 0 (3) 7	()	<u>reduce 5</u>
4	2	eof	\$ 0 Pair 2	<i>Pair</i>	reduce 3
5	1	eof	\$ 0 <i>List</i> 1	<i>List</i>	accept

state ready for reduction

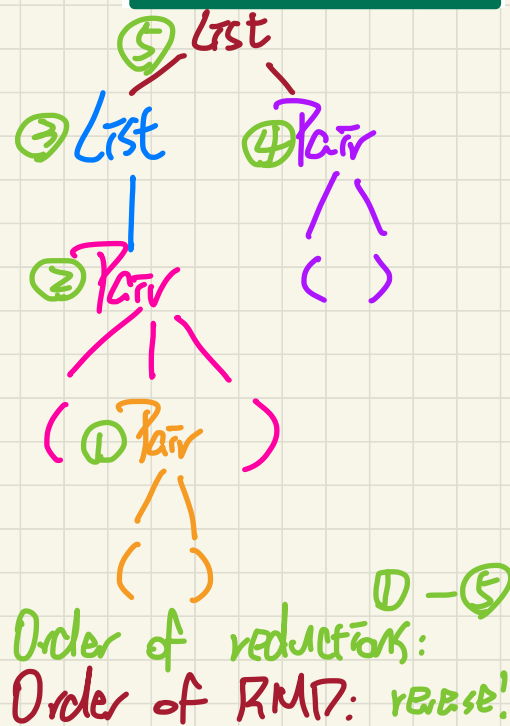
Bottom-Up Parsing: Right-Most Derivation

Parse: (()) ()

The **BUP** process corresponds to the reverse of a **RMD**.

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

Iteration	State	word	Stack	Handle	Action
initial	—	(	\$ 0	— none —	—
1	0	(	\$ 0	— none —	shift 3
2	3	(	\$ 0 (3	— none —	shift 6
3	6	)	\$ 0 (3 (6	— none —	shift 10
4	10	)	\$ 0 (3 (6) 10	()	reduce 5
5	5	)	\$ 0 (3 Pair 5	— none —	shift 8
6	8	(	\$ 0 (3 Pair 5) 8	(Pair)	reduce 4
7	2	(	\$ 0 Pair 2	Pair	reduce 3
8	1	(	\$ 0 List 1	— none —	shift 3
9	3	)	\$ 0 List 1 (3	— none —	shift 7
10	7	eof	\$ 0 List 1 (3) 7	()	reduce 5
11	4	eof	\$ 0 List 1 Pair 4	List Pair	reduce 2
12	1	eof	\$ 0 List 1	List	accept



LR(1) Items: Definition

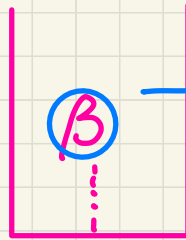
$[A \rightarrow \beta \bullet \gamma, a]$

production rule $A \rightarrow \beta \gamma$

possible states of parser

look-ahead $\in \text{Follow}(A)$

current top of stack



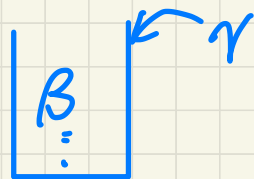
- we have already recognized β
- once we recognized γ
 $\hookrightarrow$ in a handle ready for reducing into A

LR(1) Items: Scenarios

Possibility: $[A \rightarrow \bullet \beta \gamma, a]$

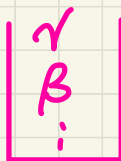
↳ initial state of parsing towards reduction to A

Partial Completion: $[A \rightarrow \beta \bullet \gamma, a]$



↳ already recognized β
still expecting to recognize γ

Completion: $[A \rightarrow \underline{\beta} \underline{\gamma} \bullet, \underline{a}]$



↓ Follow(A)
if word matches a , reduce to A

LR(1) Items: Exercise (1.1a)

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

Q. LR(1) item denoting the initial state of parsing?

$[\underline{Goal} \rightarrow \bullet List, \boxed{eof}]$

$\leftarrow Follow(Goal)$
 $\{eof\}$

Q. LR(1) item denoting the desired final state of parsing?

not necessarily
the final state

$[Pair \rightarrow () \bullet]$

$[Goal \rightarrow List \bullet, eof]$

LR(1) Items: Exercise (1.1b)

Q. Derive all LR(1) items for the production rule $A \rightarrow \beta\gamma$

- union
- set comprehension
- floating "point"

$\mathcal{I}_1$: floating positions $\downarrow \bullet$

$\rightarrow A \rightarrow \bullet \beta \gamma$

$A \rightarrow \beta \bullet \gamma$

$A \rightarrow \beta \gamma \bullet$

$\mathcal{I}_2$: $\text{Follow}(A)$

$\{ [A \rightarrow \bullet \beta \gamma, a] \mid a \in \text{Follow}(A) \}$

$\cup$
 $\{ [A \rightarrow \beta \bullet \gamma, a] \mid a \in \text{Follow}(A) \}$

$\cup$
 $\{ [A \rightarrow \beta \gamma \bullet, a] \mid a \in \text{Follow}(A) \}$

LR(1) Items: Exercise (1.2)

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

How many LR(1) items?
- possible floating
- cardinality of

Point positions: 4
Follow(Pair) = 3
12

Q. Derive all LR(1) items for the production rule $Pair \rightarrow (Pair)$

$FOLLOW(List) = \{eof, (\}$ $FOLLOW(Pair) = \{eof, (,)\}$

$[Pair \rightarrow \cdot (Pair), eof]$ $[Pair \rightarrow (\cdot Pair), eof]$ $[Pair \rightarrow (Pair \cdot), eof]$ $[Pair \rightarrow (Pair) \cdot, eof]$
 $[Pair \rightarrow \cdot (Pair), (]$ $[Pair \rightarrow (\cdot Pair), (]$ $[Pair \rightarrow (Pair \cdot), (]$ $[Pair \rightarrow (Pair) \cdot, (]$
 $[Pair \rightarrow \cdot (Pair),)]$ $[Pair \rightarrow (\cdot Pair),)]$ $[Pair \rightarrow (Pair \cdot),)]$ $[Pair \rightarrow (Pair) \cdot,)]$

LR(1) Items: Exercise (1.3)

```

1  Goal → List
2  List → List Pair
3      | Pair
4  Pair → ( Pair )
5      | ( )
    
```

FOLLOW(List) = {eof, (} **FOLLOW(Pair) = {eof, (,)}**

[Goal → • List, eof]

[Goal → List •, eof]

[List → • List Pair, eof] [List → • List Pair, (]

[List → List • Pair, eof] [List → List • Pair, (]

[List → List Pair •, eof] [List → List Pair •, (]

[List → • Pair, eof] [List → • Pair, (]

[List → Pair •, eof] [List → Pair •, (]

[Pair → • (Pair), eof] [Pair → • (Pair),] [Pair → • (Pair), (]

[Pair → (• Pair), eof] [Pair → (• Pair),] [Pair → (• Pair), (]

[Pair → (Pair •), eof] [Pair → (Pair •),] [Pair → (Pair •), (]

[Pair → (Pair) •, eof] [Pair → (Pair) •,] [Pair → (Pair) •, (]

[Pair → • (), eof] [Pair → • (), (]

[Pair → (•), eof] [Pair → (•), (]

[Pair → () •, eof] [Pair → () •, (]

LR(1) Items: Exercise (2)

0 *Goal* $\rightarrow$ *Expr*

1 *Expr* $\rightarrow$ *Term Expr'*

2 *Expr'* $\rightarrow$ + *Term Expr'*

3 | - *Term Expr'*

4 | ϵ

5 *Term* $\rightarrow$ *Factor Term'*

6 *Term'* $\rightarrow$ $\times$ *Factor Term'*

7 | $\div$ *Factor Term'*

8 | ϵ

9 *Factor* $\rightarrow$ (*Expr*)

10 | num

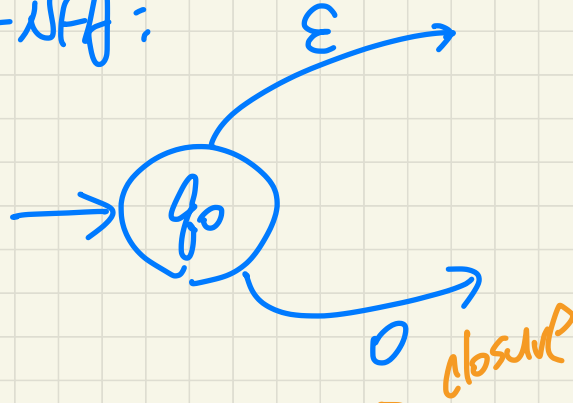
11 | name

Q. Derive all LR(1) items for the the above grammar.

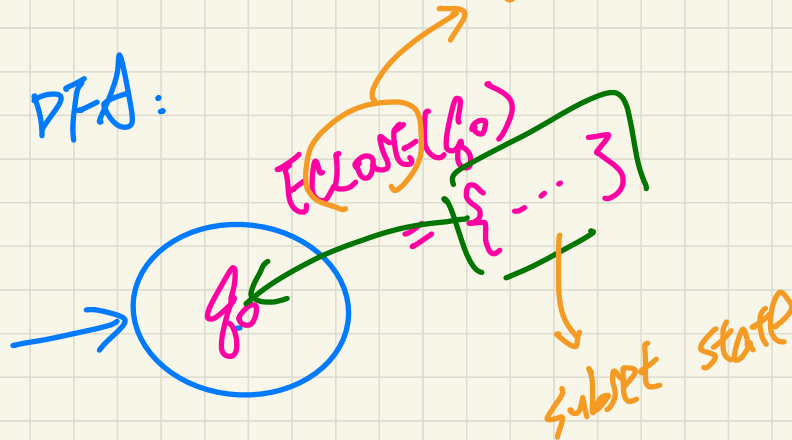
FOLLOW Set

	<i>Expr</i>	<i>Expr'</i>	<i>Term</i>	<i>Term'</i>	<i>Factor</i>
FOLLOW	eof, $_$	eof, $_$	eof, +, -, $_$	eof, +, -, $_$	eof, +, -, $\times$, $\div$, $_$

Input ϵ -NFA:



output DFA:



CC Construction: closure

```

1  ALGORITHM: closure
2  INPUT: CFG  $G = (V, \Sigma, R, S)$ , a set  $s$  of LR(1) items
3  OUTPUT: a set of LR(1) items
4  PROCEDURE:
5  lastS :=  $\emptyset$ 
6  while (lastS  $\neq$  s):
7      lastS := s
8      for  $[A \rightarrow \dots \bullet C \delta, a] \in s$ :
9          for  $C \rightarrow \gamma \in R$ :
10             for  $b \in \text{FIRST}(\delta a)$ :
11                  $s := s \cup \{ [C \rightarrow \bullet \gamma, b] \}$ 
12  return s
  
```

Handwritten notes:

- keep growing the output set s until nothing new can be added.* (points to while loop)
- all alternatives of C* (points to $C \rightarrow \gamma$)
- reducing to C* (points to C in the LR(1) item)
- 1. What has been recognized? ...* (points to C)
- 2. What's expected to be recognized next? C* (points to b)

$b \in \text{FIRST}(\delta a)$
 $\bar{\epsilon} \in \text{FIRST}(\delta)$
 Q. Why not $\bar{\epsilon} \in \text{FIRST}(\delta)$?
 $\therefore \delta$ might be nullable

Analogy: ϵ -NFA to DFA

Subset construction (with *lazy evaluation* and *epsilon closures*) produces a **DFA** transition table.

starting set (points to $\{q_0, q_1\}$)

	$d \in 0..9$	$s \in \{+, -\}$	.
$\{q_0, q_1\}$	$\{q_1, q_4\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1, q_4\}$	$\{q_1, q_4\}$	$\emptyset$	$\{q_2, q_3, q_5\}$
$\{q_1\}$	$\{q_1, q_4\}$	$\emptyset$	$\{q_2\}$
$\{q_2\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$
$\{q_2, q_3, q_5\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$
$\{q_3, q_5\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$

For example, $\delta(\{q_0, q_1\}, d)$ is calculated as follows: $[d \in 0..9]$
 $\cup \{ \text{ECLOSE}(q) \mid q \in \delta(q_0, d) \cup \delta(q_1, d) \}$

$[C \rightarrow \bullet \gamma, b]$
 new LR(1) item to be added to the closure.

set of subset states → a set of LR(1) items
CC Construction: CC₀

Calculate CC₀ of the following grammar.

Hint: Closure of the singleton set containing the parser's initial state.

1 Goal → List
 2 List → List Pair
 3 | Pair
 4 Pair → (Pair)
 5 | ()

```

1  ALGORITHM: closure
2  INPUT: CFG G = (V, Σ, R, S), a set s of LR(1) items
3  OUTPUT: a set of LR(1) items
4  PROCEDURE:
5    lastS := ∅
6    while (lastS ≠ s):
7      lastS := s
8      for [A → ... • C δ, a] ∈ s:
9        for C → γ ∈ R:
10         for b ∈ FIRST(δa):
11           s := s ∪ { [C → •γ, b] }
12  return s
  
```

Initial subset state of the parser → CC₀
 ε-NFA → CC₀
 DFA → CC₀

parser's initial state:
 $\{ [Goal \rightarrow \bullet List, \epsilon] \}$
 = input to closure

CC Construction: CC_0 Step 1

() (()) (()) ()

(0) [Goal $\rightarrow \cdot$ List, eof] initial parser state

Hint 1. How is [A $\rightarrow \cdot$ C δ , a] instantiated?

Goal $\rightarrow \cdot$ List ϵ eof

Hint 2. What are $C \rightarrow \gamma \in R$?

$\rightarrow$ List $\rightarrow$ List Pair List $\rightarrow$ Pair

Hint 3. $FIRST(\delta a) = FIRST(\epsilon eof) = FIRST(eof) = \{eof\}$

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

How should s be extended?

```
for [A  $\rightarrow \dots \cdot$  C  $\delta$ , a]  $\in$  s:
  for C  $\rightarrow \gamma \in R$ :
    for b  $\in FIRST(\delta a)$ :
      s := s  $\cup$  { [C  $\rightarrow \cdot$   $\gamma$ , b] }
```

Two new LR(1) items:

- [List $\rightarrow \cdot$ List Pair, eof]
- [List $\rightarrow \cdot$ Pair, eof]

$CC_0 = \left\{ \begin{array}{lll} [Goal \rightarrow \cdot List, eof] & [List \rightarrow \cdot List Pair, eof] & [List \rightarrow \cdot List Pair, _] \\ [List \rightarrow \cdot Pair, eof] & [List \rightarrow \cdot Pair, _] & [Pair \rightarrow \cdot _ Pair _, eof] \\ [Pair \rightarrow \cdot _ Pair _, _] & [Pair \rightarrow \cdot _ _, eof] & [Pair \rightarrow \cdot _ _, _] \end{array} \right\}$

CC Construction: CC_0 Step 2

(0) [Goal $\rightarrow \bullet$ List, eof]

(1) [List $\rightarrow \bullet$ List Pair, eof]

(2) [List $\rightarrow \bullet$ Pair, eof]

Hint 1. How is $[A \rightarrow \beta \bullet C \delta, a]$ instantiated?

Hint 2. What are $C \rightarrow \gamma \in R$? $Pair \rightarrow (Pair)$ $Pair \rightarrow ()$

Hint 3. $FIRST(\delta a) = FIRST(\epsilon eof) = FIRST(eof) = \{eof\}$

How should s be extended?

for $[A \rightarrow \dots \bullet C \delta, a] \in s$:

for $C \rightarrow \gamma \in R$:

for $b \in FIRST(\delta a)$: ✓

$s := s \cup \{ [C \rightarrow \bullet \gamma, b] \}$

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

$[Pair \rightarrow \bullet (Pair), eof]$
 $[Pair \rightarrow \bullet (), eof]$

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

CC Construction: CC_0 Step 3

(0) [Goal $\rightarrow \bullet$ List, eof]

(1) [List $\rightarrow \bullet$ List Pair, eof]

(2) [List $\rightarrow \bullet$ Pair, eof]

(3) [Pair $\rightarrow \bullet$ (Pair), eof]

(4) [Pair $\rightarrow \bullet$ (), eof]

Hint 1. How is [$\overset{\text{List}}{A} \rightarrow \overset{\text{List Pair}}{b} \bullet \overset{\text{Pair}}{C} \delta, \overset{\text{eof}}{a}$] instantiated?

Hint 2. What are $C \rightarrow \gamma \in R$?

Hint 3. $\text{FIRST}(\delta a) = \text{FIRST}(\text{Pair eof}) = \{ (\}$

How should s be extended?

for [$A \rightarrow \dots \bullet C \delta, a$] $\in s$:

for $C \rightarrow \gamma \in R$:

for $b \in \text{FIRST}(\delta a)$:

$s := s \cup \{ [C \rightarrow \bullet \gamma, b] \}$

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

[List $\rightarrow \bullet$ List Pair, (]
[List $\rightarrow \bullet$ Pair, (]

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

CC Construction: CC_0 Step 4

- (0) [Goal $\rightarrow \bullet$ List, eof] (5) [List $\rightarrow \bullet$ List Pair, (]
 (1) [List $\rightarrow \bullet$ List Pair, eof] (6) [List $\rightarrow \bullet$ Pair, (]
 (2) [List $\rightarrow \bullet$ Pair, eof]
 (3) [Pair $\rightarrow \bullet$ (Pair), eof]
 (4) [Pair $\rightarrow \bullet$ (), eof]

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

Hint 1. How is [$\underline{A} \rightarrow \underline{\beta} \bullet \underline{C} \delta, \underline{a}$] instantiated?

Hint 2. What are $C \rightarrow \gamma \in R$?

Hint 3. $FIRST(\delta a) = FIRST(\epsilon C) = \{ (\}$

How should s be extended?

```
for [A  $\rightarrow \dots \bullet$  C  $\delta$ , a]  $\in$  s:
  for C  $\rightarrow \gamma \in R$ :
    for b  $\in FIRST(\delta a)$ :
      s := s  $\cup$  { [ C  $\rightarrow \bullet \gamma$ , b ] }
```

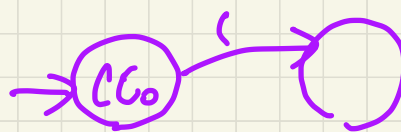
Two additional $R(1)$ items:

1. [Pair $\rightarrow \bullet$ (Pair), (]

2. [Pair $\rightarrow \bullet$ (), (]

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

CC Construction: goto



1 ALGORITHM: **goto** *source subset state*
 2 INPUT: a set S of LR(1) items, a symbol x
 3 OUTPUT: a set of LR(1) items *target subset state*
 4 PROCEDURE:
 5 $moved := \emptyset$
 6 for item $\in S$:
 7 if item = $[\alpha \rightarrow \beta \bullet x \delta, a]$ then *expecting to read x*
 8 $moved := moved \cup \{ [\alpha \rightarrow \beta x \bullet \delta, a] \}$
 9 end
 10 return closure($moved$) *x already recognized.*

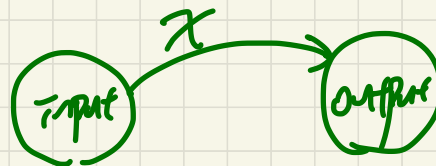
Analogy: ϵ -NFA to DFA

Subset construction (with lazy evaluation and epsilon closures) produces a DFA transition table

source

	$d \in 0..9$	$s \in \{+, -\}$	.
$\{q_0, q_1\}$	$\{q_1, q_4\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1, q_4\}$	$\{q_1, q_4\}$	$\emptyset$	$\{q_2, q_3, q_5\}$
$\{q_1\}$	$\{q_1, q_4\}$	$\emptyset$	$\{q_2\}$
$\{q_2\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$
$\{q_2, q_3, q_5\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$
$\{q_3, q_5\}$	$\{q_3, q_5\}$	$\emptyset$	$\emptyset$

For example, $\delta(\{q_0, q_1\}, d)$ is calculated as follows: $[d \in 0..9]$
 $\cup \{ \text{ECLOSE}(q) \mid q \in \delta(q_0, d) \cup \delta(q_1, d) \}$



ECLOSE $\left(\begin{array}{c} \delta(q_0, d) \\ \cup \\ \delta(q_1, d) \end{array} \right)$

CC Construction: goto

Calculate goto(cc₀, ()

i.e., "next subset state" from cc₀ taking (

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

$[Pair \rightarrow \bullet (Pair), (]$
 $[Pair \rightarrow \bullet (), eof]$
 $[Pair \rightarrow \bullet (Pair), eof]$
 $[Pair \rightarrow \bullet (), (]$

$[Pair \rightarrow (\bullet Pair), (]$
 $[Pair \rightarrow (\bullet), eof]$
 $[Pair \rightarrow (\bullet Pair), eof]$
 $[Pair \rightarrow (\bullet), (]$

closure

will trigger additional items

$cc_0 = \left\{ \begin{array}{lll} [Goal \rightarrow \bullet List, eof] & [List \rightarrow \bullet List Pair, eof] & [List \rightarrow \bullet List Pair, (] \\ [List \rightarrow \bullet Pair, eof] & [List \rightarrow \bullet Pair, (] & [Pair \rightarrow \bullet (Pair), eof] \\ [Pair \rightarrow \bullet (Pair), (] & [Pair \rightarrow \bullet (), eof] & [Pair \rightarrow \bullet (), (] \end{array} \right\}$

1 ALGORITHM: goto
 2 INPUT: a set S of LR(1) items, a symbol x
 3 OUTPUT: a set of LR(1) items
 4 PROCEDURE:
 5 $moved := \emptyset$
 6 for $item \in S$:
 7 if $item = [\alpha \rightarrow \beta \bullet x \delta, a]$ then
 8 $moved := moved \cup \{ [\alpha \rightarrow \beta x \bullet \delta, a] \}$
 9 end
 10 return closure(moved)

must be a terminal

$cc_3 = \left\{ \begin{array}{lll} [Pair \rightarrow \bullet (Pair), (] & [Pair \rightarrow (\bullet Pair), eof] & [Pair \rightarrow (\bullet Pair), (] \\ [Pair \rightarrow \bullet (), eof] & [Pair \rightarrow (\bullet), eof] & [Pair \rightarrow (\bullet), (] \end{array} \right\}$

Exercise: why the highlighted items trigger the two additional items

CC Construction: goto



Calculate **goto**(**cc₀**, **List**)

i.e., "next subset state" from **cc₀** taking * **List**

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

closure({ [Goal $\rightarrow$ List • eof],
[List $\rightarrow$ List • Pair, eof],
[List $\rightarrow$ List • Pair, (] })

cc₀ = { [Goal $\rightarrow$ • List, eof] [List $\rightarrow$ • List Pair, eof] [List $\rightarrow$ • List Pair, (]
[List $\rightarrow$ • Pair, eof] [List $\rightarrow$ • Pair, (] [Pair $\rightarrow$ • (Pair), eof]
[Pair $\rightarrow$ • (Pair), (] [Pair $\rightarrow$ • (), eof] [Pair $\rightarrow$ • (), (]

Dimension 1: Two alt. for Pair

Pair $\rightarrow$ (Pair) Dimension 2:
Pair $\rightarrow$ () FIRST(S_a)

1 ALGORITHM: goto
2 INPUT: a set S of LR(1) items, a symbol x
3 OUTPUT: a set of LR(1) items $\rightarrow$ 1. terminal / 2. variable
4 PROCEDURE:
5 moved := $\emptyset$
6 for item $\in$ S:
7 if item = [$\alpha \rightarrow \beta \bullet \delta$, a] then
8 moved := moved $\cup$ { [$\alpha \rightarrow \beta \bullet \delta$, a] }
9 end
10 return closure(moved)

cc₁ = { [Goal $\rightarrow$ List •, eof] [List $\rightarrow$ List • Pair, eof] [List $\rightarrow$ List • Pair, (]
[Pair $\rightarrow$ • (Pair), eof] [Pair $\rightarrow$ • (Pair), (] [Pair $\rightarrow$ • (), eof]
[Pair $\rightarrow$ • (), (]

CC and δ Construction: Algorithm and Exercise

1 **ALGORITHM:** *BuildCC*

2 **INPUT:** a grammar $G = (V, \Sigma, R, S)$, goal production $S \rightarrow S'$ *start var.*

3 **OUTPUT:**

4 (1) a set $CC = \{cc_0, cc_1, \dots, cc_n\}$ where $cc_i \subseteq G$'s LR(1) items

5 (2) a transition function

6 **PROCEDURE:**

7 $cc_0 := \text{closure}(\{[S^* \rightarrow \bullet S', \text{eof}]\})$

8 $CC := \{cc_0\}$

9 $\text{processed} := \{cc_0\}$

10 $\text{lastCC} := \emptyset$

11 **while** ($\text{lastCC} \neq CC$):

12 $\text{lastCC} := CC$

13 **for** cc_i s.t. $cc_i \in CC \wedge cc_i \notin \text{processed}$:

14 $\text{processed} := \text{processed} \cup \{cc_i\}$

15 **for** x s.t. $[\dots \rightarrow \dots \bullet x \dots] \in cc_i$:

16 $\text{temp} := \text{goto}(cc_i, x)$

17 **if** $\text{temp} \notin CC$ **then**

18 $CC := CC \cup \{\text{temp}\}$

19 **end**

20 $\delta := \delta \cup (cc_i, x, \text{temp})$



make a transition from CC_i via recognizing x

ready to recognize a terminal or variable

src state cc_i for state x temp

transition

1 $\text{Goal} \rightarrow \text{List}$

2 $\text{List} \rightarrow \text{List Pair}$

3 $\quad \quad | \text{Pair}$

4 $\text{Pair} \rightarrow (\text{Pair})$

5 $\quad \quad | (\quad)$

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

List

$$CC_0 = \left\{ \begin{array}{lll} [Goal \rightarrow \bullet List, eof] & [List \rightarrow \bullet List Pair, eof] & [List \rightarrow \bullet List Pair, _] \\ [List \rightarrow \bullet Pair, eof] & [List \rightarrow \bullet Pair, _] & [Pair \rightarrow \bullet _ Pair _, eof] \\ [Pair \rightarrow \bullet _ Pair _, _] & [Pair \rightarrow \bullet _ _, eof] & [Pair \rightarrow \bullet _ _, _] \end{array} \right\}$$

$$CC_1 = \left\{ \begin{array}{lll} [Goal \rightarrow List \bullet, eof] & [List \rightarrow List \bullet Pair, eof] & [List \rightarrow List \bullet Pair, _] \\ [Pair \rightarrow \bullet _ Pair _, eof] & [Pair \rightarrow \bullet _ Pair _, _] & [Pair \rightarrow \bullet _ _, eof] \\ & [Pair \rightarrow \bullet _ _, _] & \end{array} \right\}$$

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

$$CC_3 = \left\{ \begin{array}{lll} [Pair \rightarrow \bullet _ Pair _, _] & [Pair \rightarrow _ \bullet Pair _, eof] & [Pair \rightarrow _ \bullet Pair _, _] \\ [Pair \rightarrow \bullet _ _, _] & [Pair \rightarrow _ \bullet _, eof] & [Pair \rightarrow _ \bullet _, _] \end{array} \right\}$$

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

$$CC_5 = \left\{ [Pair \rightarrow _ Pair \bullet _, eof] \quad [Pair \rightarrow _ Pair \bullet _, _] \right\}$$

$$CC_6 = \left\{ \begin{array}{ll} [Pair \rightarrow \bullet _ Pair _, _] & [Pair \rightarrow _ \bullet Pair _, _] \\ [Pair \rightarrow \bullet _ _, _] & [Pair \rightarrow _ \bullet _, _] \end{array} \right\}$$

$$CC_7 = \left\{ [Pair \rightarrow _ _ \bullet, eof] \quad [Pair \rightarrow _ _ \bullet, _] \right\}$$

$$CC_8 = \left\{ [Pair \rightarrow _ Pair _ \bullet, eof] \quad [Pair \rightarrow _ Pair _ \bullet, _] \right\}$$

$$CC_9 = \left\{ [Pair \rightarrow _ Pair \bullet _, _] \right\}$$

$$CC_{10} = \left\{ [Pair \rightarrow _ _ \bullet, _] \right\}$$

$$CC_{11} = \left\{ [Pair \rightarrow _ Pair _ \bullet, _] \right\}$$

CC and δ Construction: Output 2

Transition Function

Iteration	Item	Goal	List	Pair	(	)	eof
0	CC ₀	$\emptyset$	CC ₁	CC ₂	CC ₃	$\emptyset$	$\emptyset$
1	CC ₁	$\emptyset$	$\emptyset$	CC ₄	CC ₃	$\emptyset$	$\emptyset$
	CC ₂	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	CC ₃	$\emptyset$	$\emptyset$	CC ₅	CC ₆	CC ₇	$\emptyset$
2	CC ₄	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	CC ₅	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	CC ₈	$\emptyset$
	CC ₆	$\emptyset$	$\emptyset$	CC ₉	CC ₆	CC ₁₀	$\emptyset$
	CC ₇	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	CC ₈	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	CC ₉	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	CC ₁₁	$\emptyset$
	CC ₁₀	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	CC ₁₁	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

DFA of the LR(1) Parser

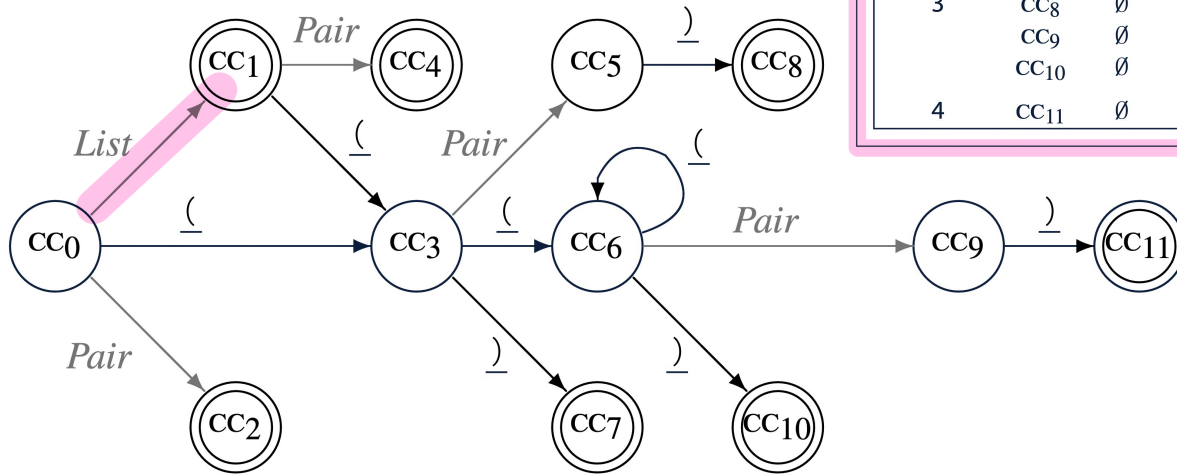


Table Construction: Algorithm

1 **ALGORITHM:** *BuildActionGotoTables*

2 **INPUT:**

3 (1) a grammar $G = (V, \Sigma, R, S)$

4 (2) goal production $S \rightarrow S'$

5 (3) a canonical collection $CC = \{CC_0, CC_1, \dots, CC_n\}$

6 (4) a transition function $\delta: CC \times \Sigma \rightarrow CC$

7 **OUTPUT:** *Action Table* & *Goto Table*

8 **PROCEDURE:**

9 for $CC_i \in CC$:

10 for item $\in CC_i$:

11 if item = $[A \rightarrow \beta \bullet x \gamma, a] \wedge \delta(CC_i, x) = CC_j$ then

12 $\rightarrow$ Action[i, x] := **shift**

13 elseif item = $[A \rightarrow \beta \bullet, a]$ then

14 Action[i, a] := **reduce** $A \rightarrow \beta$

15 elseif item = $[S \rightarrow S' \bullet, eof]$ then

16 Action[i, eof] := **accept**

17 end

18 for $v \in V$:

19 if $\delta(CC_i, v) = CC_j$ then

20 Goto[i, v] = j

21 end

*produced by
BurblCC*

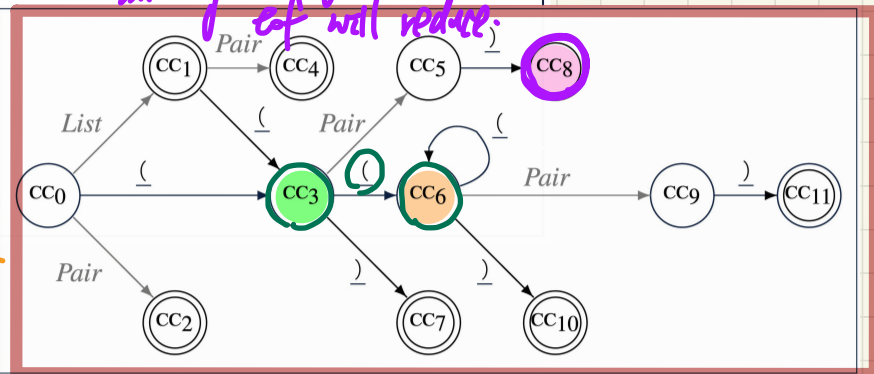
$\delta(CC_3, () = CC_6$

*CC₈ is ahead
an accepting state,
meaning eof will reduce.*

*fill in
goto table.*

State	Action Table			Goto Table	
	eof	(	)	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		

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



Bottom-Up Parsing: Discovering Ambiguities

CC₁₃ = $\left\{ \begin{array}{l} [Stmt \rightarrow \text{if expr then Stmt} \bullet, \{\underline{\text{eof}}, \underline{\text{else}}\}], \\ [Stmt \rightarrow \text{if expr then Stmt} \bullet \underline{\text{else}} Stmt, \{\text{eof}, \text{else}\}]. \end{array} \right\}$

by reading eof or else, reduce to Stmt

by reading else, we shift to

What if the current word to match is else?

What if the current word to match is a?

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

$\gamma\delta$ already recognized

shift or reduce to Stmt

shift-reduce conflict

in practice, shift will be done.

What if the current word to match is a?

some reduction

reduce-reduce conflict

must fix the grammar.