

Syntax Analysis – Part V

Finish LR(0) Parsing

Start on LR(1) Parsing

EECS 483 – Lecture 8

University of Michigan

Monday, October 2, 2006

Announcements/Reading

- ❖ Reading

- » Today 4.5, 4.7

- ❖ Project 1 signup sheet available in class

- » Grading Wed 1:30 – 3:00, Thu 1:30 – 3:30

- » If you have conflicts with both days, let me know and we'll work something out

- » Sign up for 10 min slot

- ❖ Project 2 – not ready yet, hopefully later this week

From Last Time: Shift-Reduce Parsing

$$\begin{array}{l} S \rightarrow S + E \mid E \\ E \rightarrow \text{num} \mid (S) \end{array}$$

derivation	stack	input stream	action
(1+2+(3+4))+5		(1+2+(3+4))+5	shift
(1+2+(3+4))+5	(1+2+(3+4))+5	shift
(1+2+(3+4))+5	(1	+2+(3+4))+5	reduce $E \rightarrow \text{num}$
(E+2+(3+4))+5	(E	+2+(3+4))+5	reduce $S \rightarrow E$
(S+2+(3+4))+5	(S	+2+(3+4))+5	shift
(S+2+(3+4))+5	(S+	2+(3+4))+5	shift
(S+2+(3+4))+5	(S+2	+(3+4))+5	reduce $E \rightarrow \text{num}$
(S+E+(3+4))+5	(S+E	+(3+4))+5	reduce $S \rightarrow S+E$
(S+(3+4))+5	(S	+(3+4))+5	shift
(S+(3+4))+5	(S+	(3+4))+5	shift
(S+(3+4))+5	(S+(3+4))+5	shift
(S+(3+4))+5	(S+(3	+4))+5	reduce $E \rightarrow \text{num}$

...

From Last Time: LR Parsing Table Example

We want to derive this in an algorithmic fashion

State	Input terminal					Non-terminals	
	()	id	,	\$	S	L
1	s3		s2			g4	
2	S→id	S→id	S→id	S→id	S→id		
3	s3		s2			g7	g5
4					accept		
5		s6		s8			
6	S→(L)	S→(L)	S→(L)	S→(L)	S→(L)		
7	L→S	L→S	L→S	L→S	L→S		
8	s3		s2			g9	
9	L→L,S	L→L,S	L→L,S	L→L,S	L→L,S		

From Last Time: Start State and Closure

❖ Start state

- » Augment grammar with production: $S' \rightarrow S \$$
- » Start state of DFA has empty stack: $S' \rightarrow \cdot S \$$

❖ Closure of a parser state:

- » Start with $\text{Closure}(S) = S$
- » Then for each item in S :
 - $X \rightarrow \alpha \cdot Y \beta$
 - Add items for all the productions $Y \rightarrow \gamma$ to the closure of S : $Y \rightarrow \cdot \gamma$

Closure

$S \rightarrow (L) \mid id$ $L \rightarrow S \mid L,S$

DFA start state

$S' \rightarrow \cdot S \$$

closure

$S' \rightarrow \cdot S \$$

$S \rightarrow \cdot (L)$

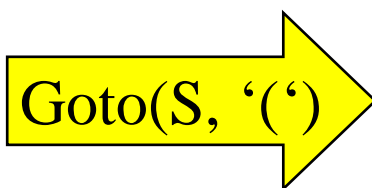
$S \rightarrow \cdot id$

- Set of possible productions to be reduced next
- Closure of a parser state, S:
 - Start with $\text{Closure}(S) = S$
 - Then for each item in S:
 - $X \rightarrow \alpha \cdot Y \beta$
 - Add items for all the productions $Y \rightarrow \gamma$ to the closure of S: $Y \rightarrow \cdot \gamma$

The Goto Operation

- ❖ Goto operation = describes transitions between parser states, which are sets of items
- ❖ Algorithm: for state S and a symbol Y
 - » If the item $[X \rightarrow \alpha \cdot Y \beta]$ is in I , then
 - » $\text{Goto}(I, Y) = \text{Closure}([X \rightarrow \alpha Y \cdot \beta])$

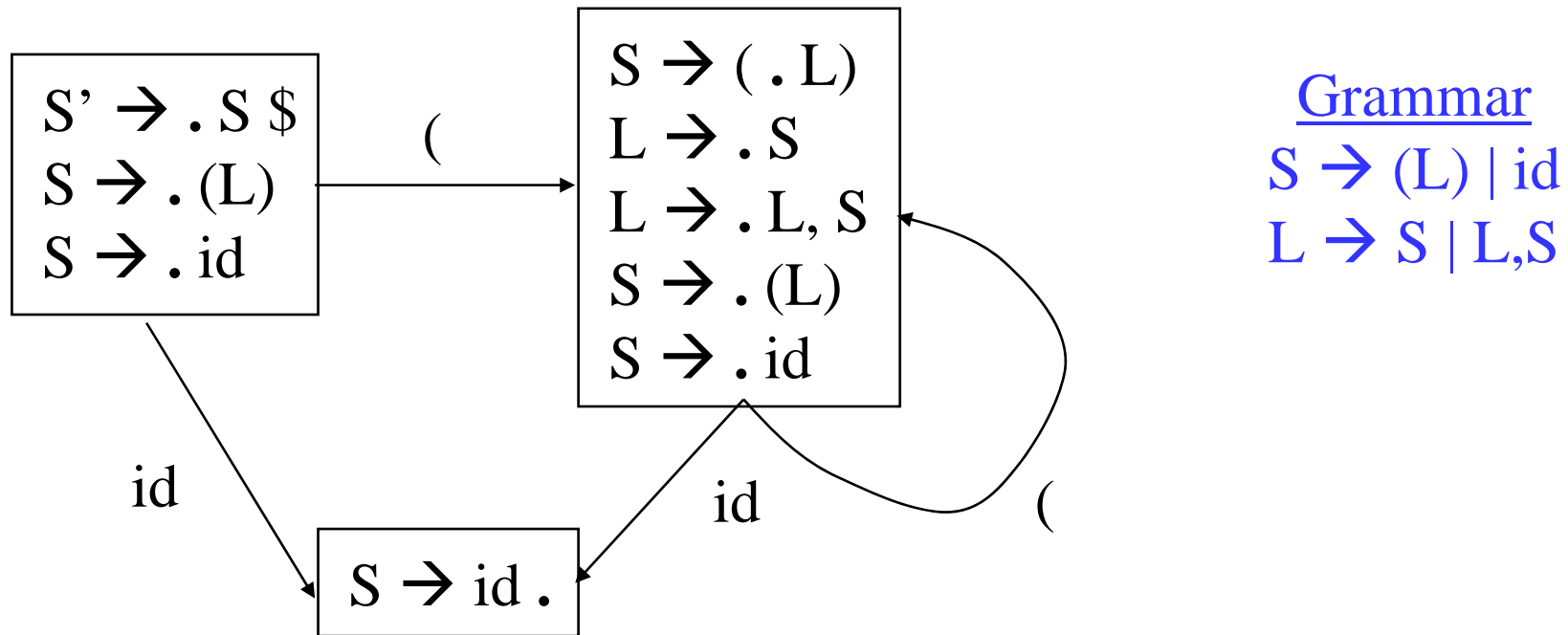
$S' \rightarrow \cdot S \$$
$S \rightarrow \cdot (L)$
$S \rightarrow \cdot \text{id}$



$\text{Goto}(S, '(')$

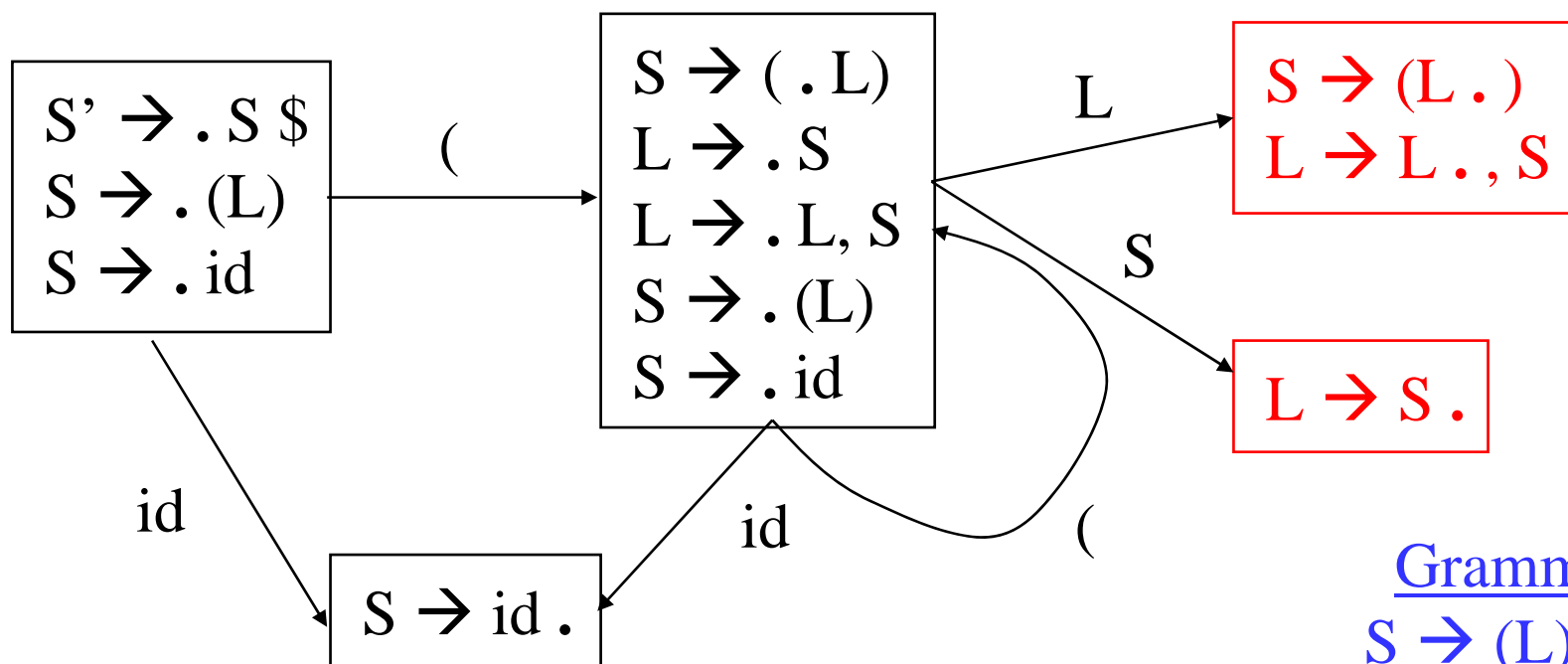
$\text{Closure}(\{ S \rightarrow (\cdot L) \})$

Goto: Terminal Symbols



In new state, include all items that have appropriate input symbol just after dot, advance dot in those items and take closure

Goto: Non-terminal Symbols



Grammar
 $S \rightarrow (L) \mid id$
 $L \rightarrow S \mid L, S$

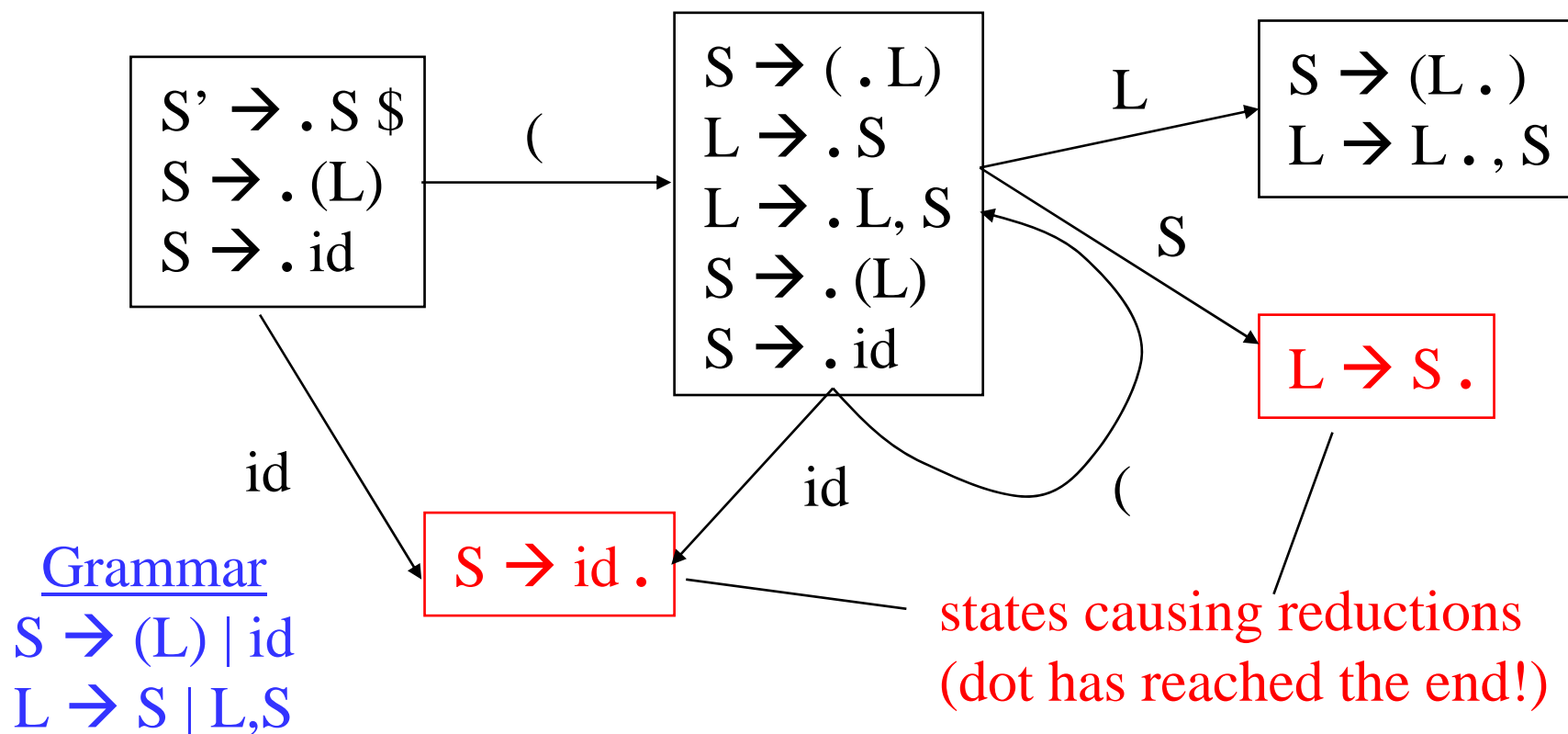
same algorithm for transitions on non-terminals

Class Problem

$$\begin{array}{l} E' \rightarrow E \\ E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid \text{id} \end{array}$$

1. If $I = \{ [E' \rightarrow \cdot E] \}$, then $\text{Closure}(I) = ??$
2. If $I = \{ [E' \rightarrow E \cdot], [E \rightarrow E \cdot + T] \}$, then $\text{Goto}(I, +) = ??$

Applying Reduce Actions



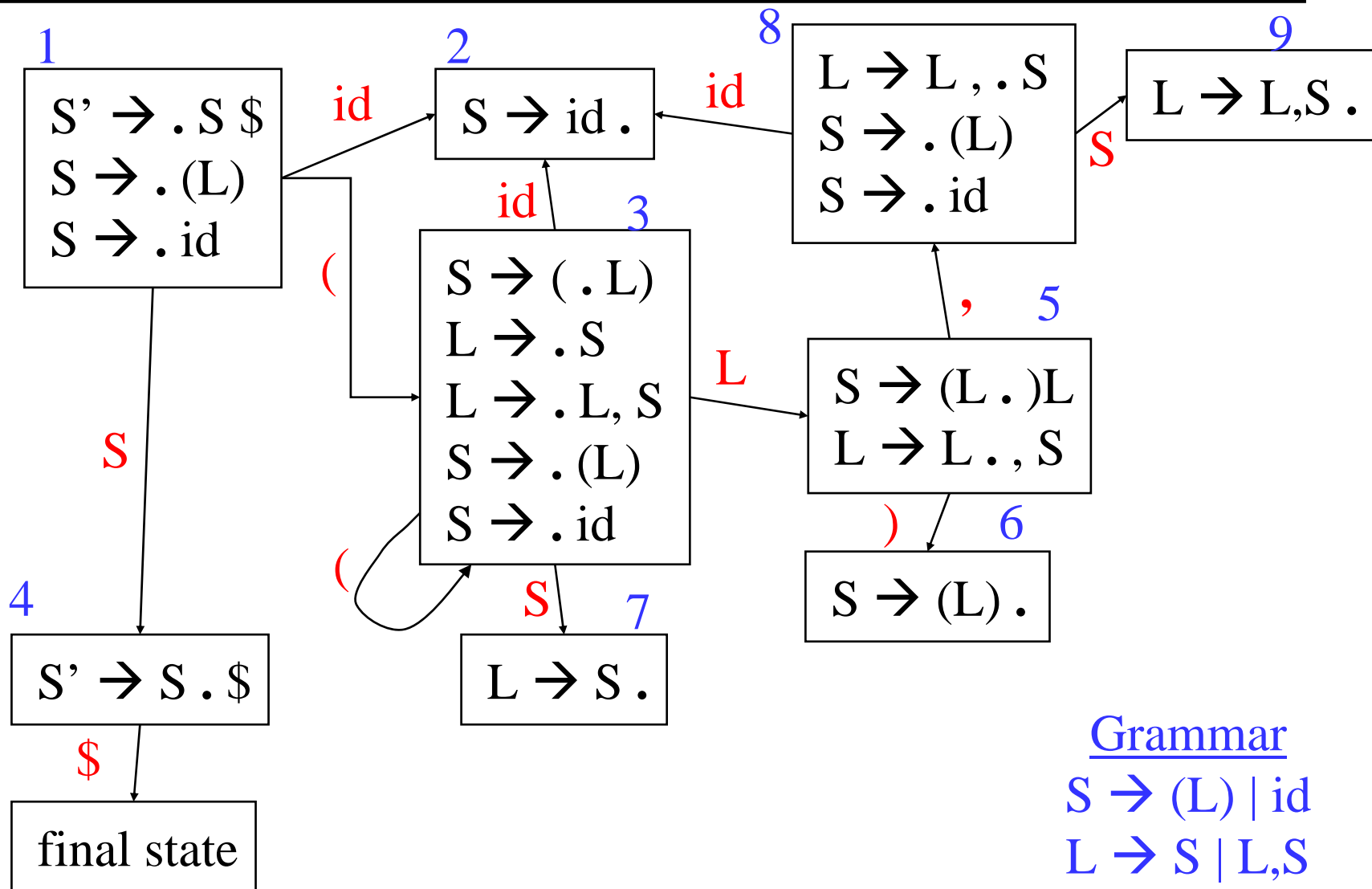
Pop RHS off stack, replace with LHS X ($X \rightarrow \beta$),
then rerun DFA (e.g., (x))

Reductions

- ❖ On reducing $X \rightarrow \beta$ with stack $\alpha\beta$
 - » Pop β off stack, revealing prefix α and state
 - » Take single step in DFA from top state
 - » Push X onto stack with new DFA state
- ❖ Example

derivation	stack	input	action
$((a),b) \leftarrow$	1 (3 (3	a),b)	shift, goto 2
$((a),b) \leftarrow$	1 (3 (3 a 2),b)	reduce $S \rightarrow id$
$((S),b) \leftarrow$	1 (3 (3 S 7),b)	reduce $L \rightarrow S$

Full DFA



Parsing Example ((a),b)

$S \rightarrow (L) \mid id$

$L \rightarrow S \mid L,S$

derivation	stack	input	action
((a),b) ←	1	((a),b)	shift, goto 3
((a),b) ←	1(3	(a),b)	shift, goto 3
((a),b) ←	1(3(3	a),b)	shift, goto 2
((a),b) ←	1(3(3a2),b)	reduce $S \rightarrow id$
((S),b) ←	1(3(3(S7),b)	reduce $L \rightarrow S$
((L),b) ←	1(3(3(L5),b)	shift, goto 6
((L),b) ←	1(3(3L5)6	,b)	reduce $S \rightarrow (L)$
(S,b) ←	1(3S7	,b)	reduce $L \rightarrow S$
(L,b) ←	1(3L5	,b)	shift, goto 8
(L,b) ←	1(3L5,8	b)	shift, goto 9
(L,b) ←	1(3L5,8b2)	reduce $S \rightarrow id$
(L,S) ←	1(3L8,S9)	reduce $L \rightarrow L,S$
(L) ←	1(3L5)	shift, goto 6
(L) ←	1(3L5)6		reduce $S \rightarrow (L)$
S ←	1S4	\$	done

Building the Parsing Table

- ❖ States in the table = states in the DFA
- ❖ For transition $S \rightarrow S'$ on terminal C :
 - » $\text{Table}[S,C] += \text{Shift}(S')$
- ❖ For transition $S \rightarrow S'$ on non-terminal N :
 - » $\text{Table}[S,N] += \text{Goto}(S')$
- ❖ If S is a reduction state $X \rightarrow \beta$ then:
 - » $\text{Table}[S,*] += \text{Reduce}(X \rightarrow \beta)$

Computed LR Parsing Table

State	Input terminal					Non-terminals	
	()	id	,	\$	S	L
1	s3		s2			g4	
2	S→id	S→id	S→id	S→id	S→id		
3	s3		s2			g7	g5
4					accept		
5		s6		s8			
6	S→(L)	S→(L)	S→(L)	S→(L)	S→(L)		
7	L→S	L→S	L→S	L→S	L→S		
8	s3		s2			g9	
9	L→L,S	L→L,S	L→L,S	L→L,S	L→L,S		

blue = shift

red = reduce

LR(0) Summary

- ❖ LR(0) parsing recipe:
 - » Start with LR(0) grammar
 - » Compute LR(0) states and build DFA:
 - Use the closure operation to compute states
 - Use the goto operation to compute transitions
 - » Build the LR(0) parsing table from the DFA
- ❖ This can be done automatically

Class Problem

Generate the DFA for the following grammar

$S \rightarrow E + S \mid E$

$E \rightarrow \text{num}$

LR(0) Limitations

- ❖ An LR(0) machine only works if states with reduce actions have a single reduce action
 - » Always reduce regardless of lookahead
- ❖ With a more complex grammar, construction gives states with shift/reduce or reduce/reduce conflicts
- ❖ Need to use lookahead to choose

OK

$L \rightarrow L, S.$

shift/reduce

$L \rightarrow L, S.$
 $S \rightarrow S., L$

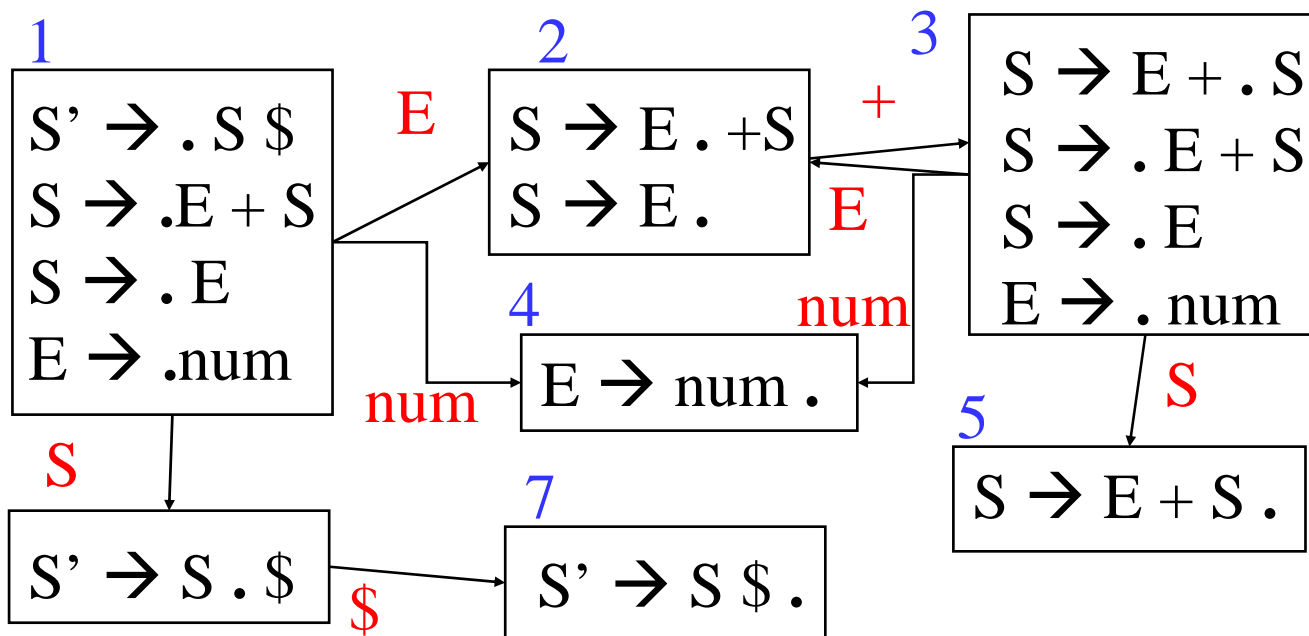
reduce/reduce

$L \rightarrow S, L.$
 $L \rightarrow S.$

A Non-LR(0) Grammar

- ❖ Grammar for addition of numbers
 - » $S \rightarrow S + E \mid E$
 - » $E \rightarrow \text{num}$
- ❖ Left-associative version is LR(0)
- ❖ Right-associative is **not LR(0)** as you saw with the previous class problem
 - » $S \rightarrow E + S \mid E$
 - » $E \rightarrow \text{num}$

LR(0) Parsing Table



Grammar
 $S \rightarrow E + S \mid E$
 $E \rightarrow \text{num}$

Shift or
 reduce
 in state 2?

	num	+	\$	E	S
1	s4			g2	g6
2	$S \rightarrow E$	s3/ $S \rightarrow E$	$S \rightarrow E$		

Solve Conflict With Lookahead

- ❖ 3 popular techniques for employing lookahead of 1 symbol with bottom-up parsing
 - » SLR – Simple LR
 - » LALR – LookAhead LR
 - » LR(1)
- ❖ Each as a different means of utilizing the lookahead
 - » Results in different processing capabilities

SLR Parsing

- ❖ SLR Parsing = Easy extension of LR(0)
 - » For each reduction $X \rightarrow \beta$, look at next symbol C
 - » Apply reduction only if C is in FOLLOW(X)
- ❖ SLR parsing table eliminates some conflicts
 - » Same as LR(0) table except reduction rows
 - » Adds reductions $X \rightarrow \beta$ only in the columns of symbols in FOLLOW(X)

Example: FOLLOW(S) = { $\$$ }

Grammar
 $S \rightarrow E + S \mid E$
 $E \rightarrow \text{num}$

	num	+	\$	E	S
1	s4			g2	g6
2		s3		$S \rightarrow E$	

SLR Parsing Table

- ❖ Reductions do not fill entire rows as before
- ❖ Otherwise, same as LR(0)

Grammar
 $S \rightarrow E + S \mid E$
 $E \rightarrow \text{num}$

	num	+	\$	E	S
1	s4			g2	g6
2		s3	$S \rightarrow E$		
3	s4			g2	g5
4		$E \rightarrow \text{num}$	$E \rightarrow \text{num}$		
5			$S \rightarrow E + S$		
6			s7		
7			accept		

Class Problem

Consider:

$S \rightarrow L = R$

$S \rightarrow R$

$L \rightarrow *R$

$L \rightarrow \text{ident}$

$R \rightarrow L$

Think of L as l-value, R as r-value, and * as a pointer dereference

When you create the states in the SLR(1) DFA,
2 of the states are the following:

$S \rightarrow L . = R$ $R \rightarrow L .$
--

$S \rightarrow R .$

Do you have any shift/reduce conflicts? (Not as easy as it looks)

LR(1) Parsing

- ❖ Get as much as possible out of 1 lookahead symbol parsing table
- ❖ LR(1) grammar = recognizable by a shift/reduce parser with 1 lookahead
- ❖ LR(1) parsing uses similar concepts as LR(0)
 - » Parser states = set of items
 - » LR(1) item = LR(0) item + lookahead symbol possibly following production
 - LR(0) item: $S \rightarrow . S + E$
 - LR(1) item: $S \rightarrow . S + E \text{ , } \underline{+}$
 - Lookahead only has impact upon REDUCE operations, apply when lookahead = next input

LR(1) States

- ❖ LR(1) state = set of LR(1) items
- ❖ LR(1) item = $(X \rightarrow \alpha \cdot \beta, y)$
 - » Meaning: α already matched at top of the stack, next expect to see β y
- ❖ Shorthand notation
 - » $(X \rightarrow \alpha \cdot \beta, \{x_1, \dots, x_n\})$
 - » means:
 - $(X \rightarrow \alpha \cdot \beta, x_1)$
 - ...
 - $(X \rightarrow \alpha \cdot \beta, x_n)$
- ❖ Need to extend closure and goto operations

$S \rightarrow S \cdot + E$	$+, \$$
$S \rightarrow S + \cdot E$	num

LR(1) Closure

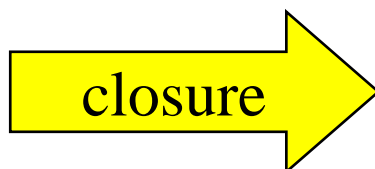
- ❖ LR(1) closure operation:
 - » Start with $\text{Closure}(S) = S$
 - » For each item in S :
 - $X \rightarrow \alpha \cdot Y \beta, z$
 - and for each production $Y \rightarrow \gamma$, add the following item to the closure of S : $Y \rightarrow \cdot \gamma, \text{FIRST}(\beta z)$
 - » Repeat until nothing changes
- ❖ Similar to LR(0) closure, but also keeps track of lookahead symbol

LR(1) Start State

- ❖ Initial state: start with $(S' \rightarrow \cdot S, \$)$, then apply closure operation
- ❖ Example: sum grammar

$S' \rightarrow S \$$
 $S \rightarrow E + S \mid E$
 $E \rightarrow \text{num}$

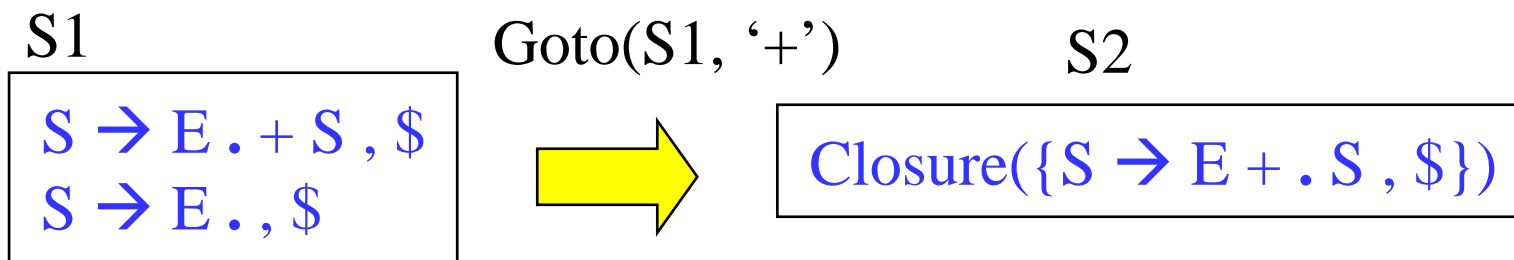
$S' \rightarrow \cdot S, \$$



$S' \rightarrow \cdot S, \$$
 $S \rightarrow \cdot E + S, \$$
 $S \rightarrow \cdot E, \$$
 $E \rightarrow \cdot \text{num}, +, \$$

LR(1) Goto Operation

- ❖ LR(1) goto operation = describes transitions between LR(1) states
- ❖ Algorithm: for a state S and a symbol Y (as before)
 - » If the item $[X \rightarrow \alpha \cdot Y \beta]$ is in I , then
 - » $\text{Goto}(I, Y) = \text{Closure}([X \rightarrow \alpha Y \cdot \beta])$



Grammar:

$S' \rightarrow S\$$

$S \rightarrow E + S \mid E$

$E \rightarrow \text{num}$

Class Problem

1. Compute: $\text{Closure}(I = \{S \rightarrow E + \cdot S, \$\})$

2. Compute: $\text{Goto}(I, \text{num})$

3. Compute: $\text{Goto}(I, E)$

$S' \rightarrow S \$$

$S \rightarrow E + S \mid E$

$E \rightarrow \text{num}$