



CSCI 742 - Compiler Construction

Lecture 19

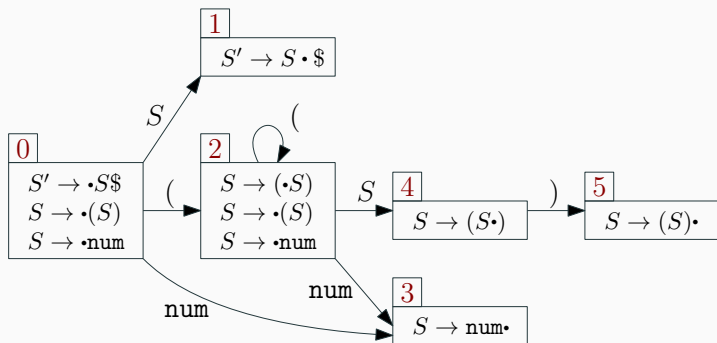
SLR and LR(1) Parsing

Instructor: Hossein Hojjat

March 8, 2017

LR(0) Automaton Example

- Consider the grammar $S \rightarrow (S) \mid \text{num}$



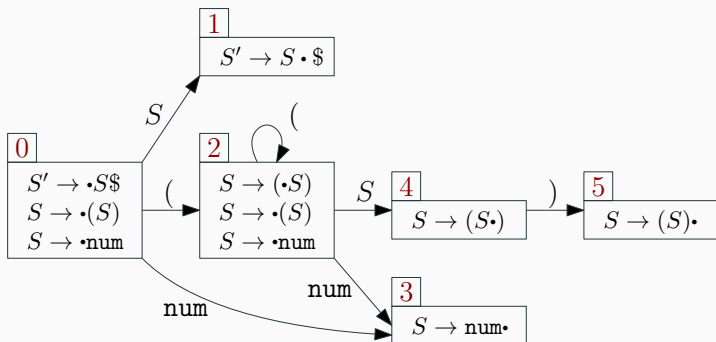
Creating Parse Tables

For each state:

- Transition to another state using a terminal symbol is a **shift** to that state
- Transition to another state using a non-terminal is a **goto** to that state
- If there is a single item $A \rightarrow \alpha \cdot$ in the state **reduce** with that production for all terminals

Building Parse Table Example

	()	num	\$	S
0	<i>s2</i>		<i>s3</i>		<i>g1</i>
1	accept				
2	<i>s2</i>		<i>s3</i>		<i>g4</i>
3	$r(S \rightarrow \text{num})$	$r(S \rightarrow \text{num})$	$r(S \rightarrow \text{num})$	$r(S \rightarrow \text{num})$	
4	<i>s5</i>				
5	$r(S \rightarrow (S))$	$r(S \rightarrow (S))$	$r(S \rightarrow (S))$	$r(S \rightarrow (S))$	



LR(0) Limitations

- LR(0) only works if states with reduce actions have a single reduce action

$$E \rightarrow T \bullet$$

- In those states it always reduce without looking at lookahead
- LR(0) is vulnerable to unnecessary conflicts
- Shift/Reduce Conflicts (may reduce too soon in some cases)

$$\begin{array}{l} E \rightarrow E \bullet + T \\ S \rightarrow E \bullet \end{array}$$

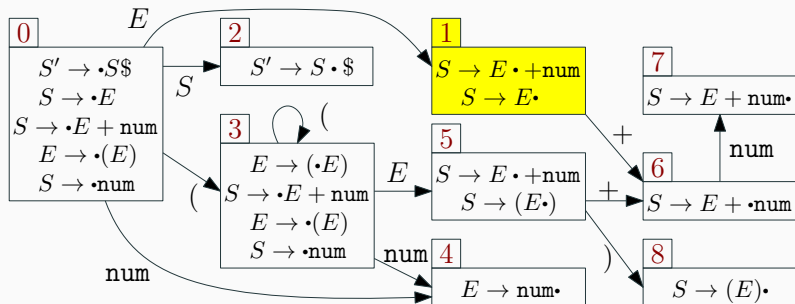
- Reduce/Reduce Conflicts

$$\begin{array}{l} E \rightarrow \text{num} \bullet \\ T \rightarrow \text{num} \bullet \end{array}$$

LR(0) Parsing Table With Conflicts

	()	+	num	\$	S	E
0	<i>s3</i>			<i>s4</i>		<i>g2</i>	<i>g1</i>
1	<i>r1</i>	<i>r1</i>	<i>r1/s6</i>	<i>r1</i>	<i>r1</i>		
2	accept						
3	<i>s3</i>			<i>s4</i>		<i>g5</i>	
4	<i>r4</i>	<i>r4</i>	<i>r4</i>	<i>r4</i>	<i>r4</i>		
5		<i>s8</i>	<i>s6</i>				
6				<i>s7</i>			
7	<i>r2</i>	<i>r2</i>	<i>r2</i>	<i>r2</i>	<i>r2</i>		
8	<i>r3</i>	<i>r3</i>	<i>r3</i>	<i>r3</i>	<i>r3</i>		

r1 $S \rightarrow E$
r2 $E \rightarrow E + \text{num}$
r3 $E \rightarrow (E)$
r4 $E \rightarrow \text{num}$



- Simple LR parsing (SLR) is a simple extension of LR(0) parsing
- For each reduction $A \rightarrow \gamma \cdot$ look at the lookahead symbol c
- Apply reduction only if c is in FOLLOW(A)

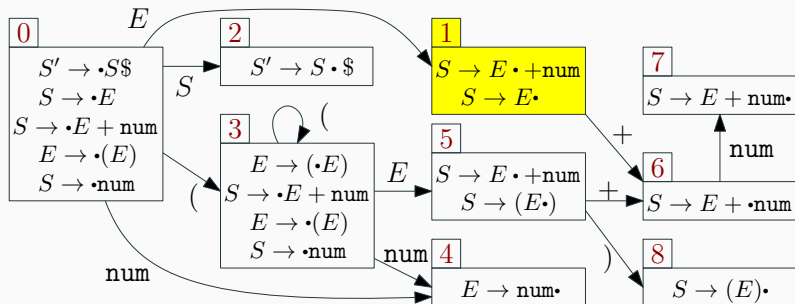
SLR Parsing Table

- Eliminates some conflicts
- Same as LR(0) table except reduction rows
- Reductions do not fill entire rows
- Add reductions $A \rightarrow \gamma \cdot$ only in the columns of symbols in FOLLOW(A)

LR(0) Parsing Table

	()	+	num	\$	S	E
0	s3			s4		g2	g1
1	r1	r1	r1/s6	r1	r1		
2	accept						
3	s3			s4		g5	
4	r4	r4	r4	r4	r4		
5		s8	s6				
6				s7			
7	r2	r2	r2	r2	r2		
8	r3	r3	r3	r3	r3		

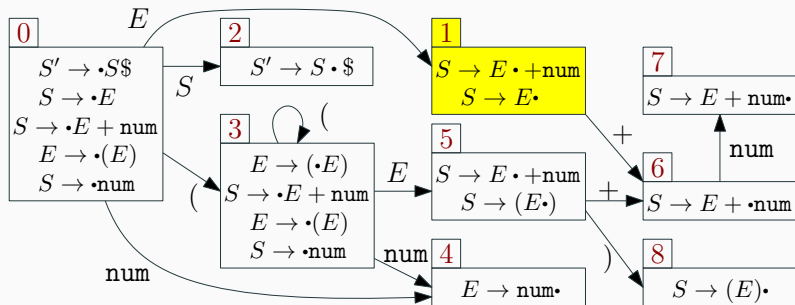
r1 $S \rightarrow E$
 r2 $E \rightarrow E + \text{num}$
 r3 $E \rightarrow (E)$
 r4 $E \rightarrow \text{num}$



SLR Parsing Table

	()	+	num	\$	S	E
0	s3			s4		g2	g1
1			s6		r1		
2					accept		
3	s3			s4			g5
4		r4	r4		r4		
5		s8	s6				
6				s7			
7		r2	r2		r2		
8		r3	r3		r3		

r1 $S \rightarrow E$
 r2 $E \rightarrow E + \text{num}$
 r3 $E \rightarrow (E)$
 r4 $E \rightarrow \text{num}$



LR(1) Parsing

- **Idea:** 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 LR(1) items
- LR(1) item = LR(0) item + lookahead symbols possibly following production
- LR(0) item: $S \rightarrow \cdot S + E$
- LR(1) item: $S \rightarrow \cdot S + E$, +
- Lookahead only has impact on 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 βy
- Shorthand notation: $(X \rightarrow \alpha \cdot \beta, \{x_1, \dots, x_n\})$ means:
 - $(X \rightarrow \alpha \cdot \beta, x_1)$
 - \dots
 - $(X \rightarrow \alpha \cdot \beta, x_n)$
- Need to extend closure and goto operations

Similar to LR(0) closure, but also keeps track of lookahead symbol

If L is a set of items, $\text{CLOSURE}(L)$ is the set of items such that:

- every item in L is in $\text{CLOSURE}(L)$
- if item $(X \rightarrow \alpha \cdot Y\beta, z)$ is in $\text{CLOSURE}(L)$ and $Y \rightarrow \gamma$ is a production then $(Y \rightarrow \cdot\gamma, \text{FIRST}(\beta z))$ is also in $\text{CLOSURE}(L)$

LR(1) Start State

Initial state: start with $(S' \rightarrow \cdot S, \$)$, then apply closure operation

Goto is analogous to goto in LR(0) parsing

Goto(L, X)

$I = \emptyset$

for any item $[A \rightarrow \alpha \cdot X\beta, x]$ in L

$I = I \cup \{[A \rightarrow \alpha X \cdot \beta, x]\}$

return CLOSURE(I)

Construct the LR(1) automaton for the following grammar:

$$S' \rightarrow S\$$$

$$S \rightarrow E + S \mid E$$

$$E \rightarrow \text{num}$$