CSCI 742 - Compiler Construction

Lecture 15

LR(0) Parsing

Instructor: Hossein Hojjat

February 19, 2018

Recap: Action Selection Problem

- Given stack σ and look-ahead symbol b, should parser:
- **shift** b onto the stack (making it σb)
- **reduce** some production $X \to \gamma$ assuming stack has the form $\alpha\gamma$ (making it αX)

Parser States

- **Goal:** know which reductions are legal at any given point
- **Idea:** summarize all possible stacks σ as a finite parser state
- Parser state is computed by a DFA that reads in the stack σ
- Accept states of DFA: unique reduction

We encode the DFA into parsing tables:

- rows: states of parser
- columns: token(s) of lookahead
- entries: action of parser

LR Parsing Engine

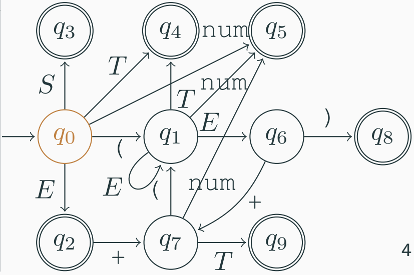
- Use stack with alternating symbols and states
  - For example: (3 a 10 $+$ 5 (red = state numbers)
- Use parsing table to:
  - Determine what action to apply (shift/reduce)
  - Determine next state
- The parser actions can be precisely determined from the table

LR Parsing Table Example

	(	)	+	num	$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \texttt{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) $	

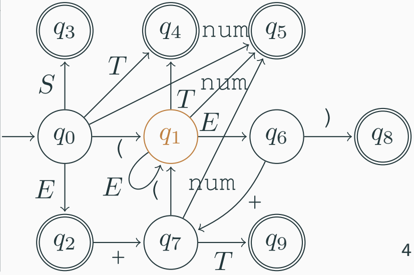


LR Parsing Table Example

	(	)	+	num	\$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \text{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) \$	shift 1
0 (1	num) \$	

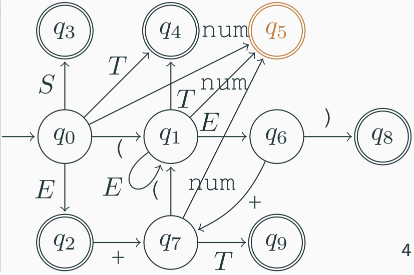


LR Parsing Table Example

	(	)	+	num	$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \mathtt{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) $	shift 1
0 (1	num) $	shift 5
0 (1 num 5	) $	

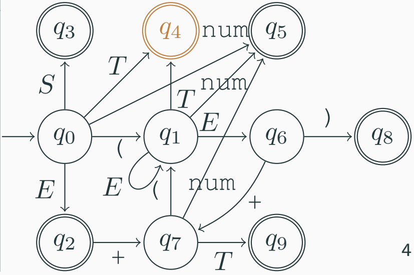


LR Parsing Table Example

	(	)	+	num	$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \texttt{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) $	shift 1
0 (1	num) $	shift 5
0 (1 num 5	) $	reduce 4
0 (1 T 4	) $	

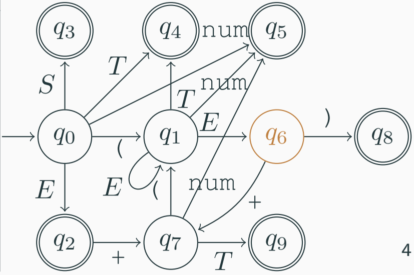


LR Parsing Table Example

	(	)	+	num	$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \text{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) $	shift 1
0 (1	num) $	shift 5
0 (1 num 5	) $	reduce 4
0 (1 T 4	) $	reduce 2
0 (1 E 6	) $	

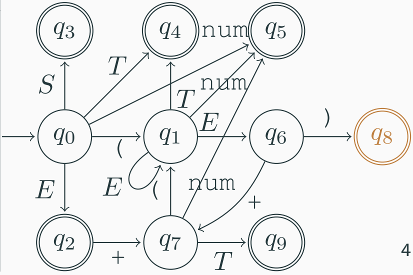


LR Parsing Table Example

	(	)	+	num	\$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \texttt{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) \$	shift 1
0 (1	num) \$	shift 5
0 (1 num 5	) \$	reduce 4
0 (1 T 4	) \$	reduce 2
0 (1 E 6	) \$	shift 8
0 (1 E 6) 8	\$	

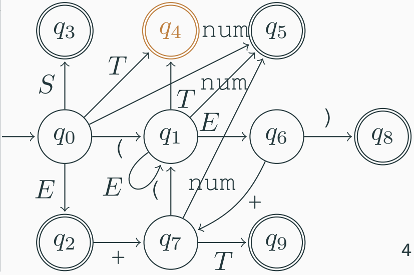


LR Parsing Table Example

	(	)	+	num	\$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \texttt{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) \$	shift 1
0 (1	num) \$	shift 5
0 (1 num 5	) \$	reduce 4
0 (1 T 4	) \$	reduce 2
0 (1 E 6	) \$	shift 8
0 (1 E 6) 8	\$	reduce 5
0 T 4	\$	

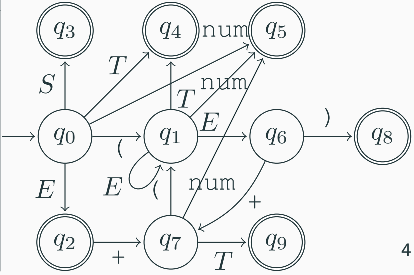


LR Parsing Table Example

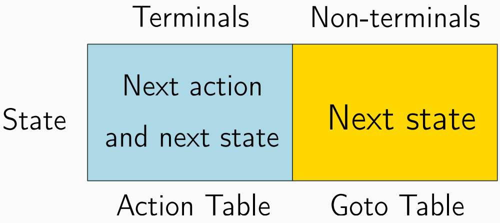
	(	)	+	num	\$	E	S	T
0	$s1$			$s5$		$g2$	$g3$	$g4$
1	$s1$			$s5$		$g6$		$g4$
2			$s7$		$r1$			
3					acc			
4		$r2$	$r2$		$r2$			
5		$r4$	$r4$		$r4$			
6		$s8$	$s7$					
7	$s1$			$s5$				$g9$
8		$r5$	$r5$		$r5$			
9		$r3$	$r3$		$r3$			

r1	$S \to E\$$
r2	$E \to T$
r3	$E \to E + T$
r4	$T \to \text{num}$
r5	$T \to (E)$

Stack	Input	Action
0	(num) \$	shift 1
0 (1	num) \$	shift 5
0 (1 num 5	) \$	reduce 4
0 (1 T 4	) \$	reduce 2
0 (1 E 6	) \$	shift 8
0 (1 E 6) 8	\$	reduce 5
0 T 4	\$	…



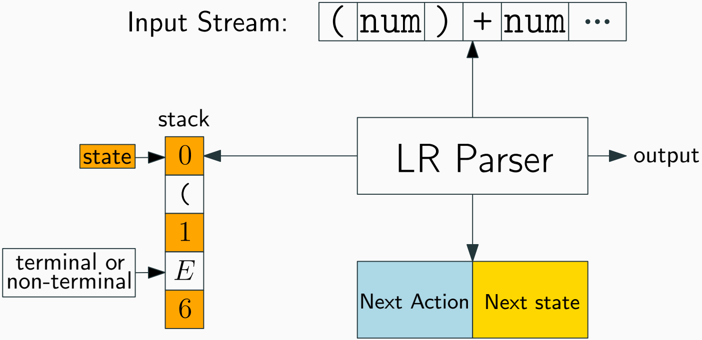
LR Parsing Table



Algorithm:

- Look at entry for current state S and input terminal C
- If Table[S,C] = s(S') then shift:
  - push(C), push(S')
- If Table[S,C] = $X \rightarrow \alpha$ then reduce:
  - pop($2 \times |\alpha|$), S'= top(), push(X), push(Table[S',X])

Model of LR Parser



LR(0) Parsing Tables

- **L**eft-to-right scanning, **R**ight-most derivation, "zero" look-ahead tokens
- Too weak to handle most language grammars

Algorithm for building LR(0) parsing tables:

1. Compute parser states
2. Build a DFA to describe the transition between states
3. Use the DFA to build the parsing table

LR(0) States

- Each LR(0) state is a set of LR(0) items
- An LR(0) item is a production from the language with a separator somewhere in the RHS
- $X \to \alpha \bullet \beta$ says that
  - parser is looking for an X
  - it has an α on top of the stack
  - expects to find in the input a string derived from β

Example: LR(0) Items

r1	$S \rightarrow E\$$
r2	$E \rightarrow T$
r3	$E \rightarrow E + T$
r4	$T \rightarrow \mathtt{num}$
r5	$T \rightarrow (E)$

LR(0) Items ⇒

1	$S \rightarrow \bullet E\$$
2	$S \rightarrow E \bullet \$$
3	$S \rightarrow E\$\bullet$
4	$E \rightarrow \bullet T$
5	$E \rightarrow T\bullet$
6	$E \rightarrow \bullet E + T$
7	$E \rightarrow E \bullet + T$
8	$E \rightarrow E + \bullet T$
9	$E \rightarrow E + T\bullet$
10	$T \rightarrow \bullet \mathtt{num}$
11	$T \rightarrow \mathtt{num}\bullet$
12	$T \rightarrow \bullet(E)$
13	$T \rightarrow (\bullet E)$
14	$T \rightarrow (E\bullet)$
15	$T \rightarrow (E)\bullet$

LR(0) Shift/Reduce Items

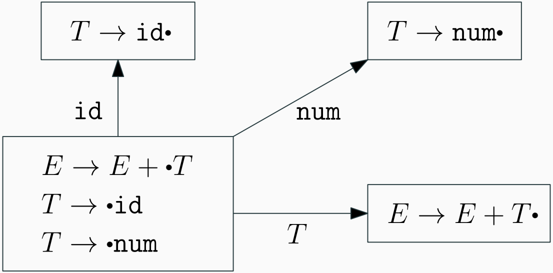
$N \to \alpha \bullet \beta$ Shift Item

$N \to \alpha\beta\bullet$ Reduce Item

LR(0) Automaton

An LR(0) automaton has:

- **states:** sets of LR(0) items
- **transitions:** label by grammar terminals or non-terminals



Construction of Automaton

- Augment grammar with production $S' \to S\$$
- Start state of automaton has empty stack $S' \to \bullet S\$$

To construct the automaton from the start state we need two functions:

- CLOSURE(L) to build its states
- GOTO(L, X) to determine its transitions

Begin with $\{S' \to \bullet S\$\}$, take the closure, and then keep applying GOTO

CLOSURE

If L is a set of items, CLOSURE(L) is the set of items such that:

- every item in L is in CLOSURE(L)
- if item $X \to \alpha \bullet Y\beta$ is in CLOSURE(L) and
  $Y \to \gamma$ is a production then
  $Y \to \bullet\gamma$
  is also in CLOSURE(L)

Exercise

- For the grammar

$$\begin{aligned} S &\to E \\ E &\to E + T \\ &\mid \; T \\ T &\to \texttt{num} \end{aligned}$$

Compute the CLOSURE of the set of items $\{S \to \bullet E\}$

GOTO

If L is a set of items and X is a grammar symbol and $Y \to \alpha \bullet X\beta \in L$ then GOTO(L, X) is the CLOSURE of the set of all items $Y \to \alpha X \bullet \beta$

Exercise

- For the grammar

$$\begin{aligned} S &\rightarrow E \\ E &\rightarrow E + T \\ &\mid\ T \\ T &\rightarrow \mathrm{num} \end{aligned}$$

Compute the state that can be reached from the LR(0) state $\{S \rightarrow E \bullet\ ,\ E \rightarrow E \bullet + T\}$ on symbol +