LR-Parsing using Action – GoTo Table

ACTION (state, *lookahead-symbol*) = Shift-N or Reduce-N

Shift-N (sN): which state symbol N to be pushed to the stack with the lookahead input symbol.

Push a lookahead-input-symbol and a state N to the stack.

- A state-symbol N will be pushed to the stack with the *lookahead input symbol* where
  LR[state, lookahead] = sN.
- Push lookahead to the stack, then
- Push state-N to the stack.

Reduce-N (rN): Replace RHD of a rule with LHD and push a new state-symbol.

- Which LHD of a rule N will replace its RHD (i.e. the handle) on the stack?
  - Remove the elements from the stack: RHD of the rule N and with their state-symbol(s)
  - Push LHD of a rule N
- Which new state-symbol to be pushed to the stack? -- use two symbols on the top of the stack
  - The new stack top (= LHD of a rule N) = top
  - A state-symbol below the top = previous-state
  - GOTO(pre-state, top) = new-state-symbol
  - Push new-state-symbol to the stack.

i.e. handle is removed from stack and new NonTerminal is pushed to stack.

GOTO (state-N, symbol): Decide which state-symbol to be pushed to stack after reduction.

Which state-symbol to be pushed to stack after reduction?
i.e. handle is removed from stack and new NonTerminal is pushed to stack. (RHS is replaced with LHS

- GOTO(state-N, symbol) = state-X
- Push state-X to the stack

LR Parser Table – generated from the grammar below using a tool: algorithm at Handout LR-Parsing

state symbol

	Action : terminal symbol						Goto: Nonterminal symbol		
State	a	+	*	(	)	$	E	T	F
0	S5		S4				1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Example: use the LR Parser table above.

1. E → E + T
2. E → T
3. T → T * F
4. T → F
5. F → (E)
6. F → a

Stack	Input	Lookahead symbol		Action	Sentential Forms
0	a + a * a $	a	LR(0, a) = S5	Shift 5	a + a * a $
0a5	+ a * a $	+	LR(5, +) = R6 LR(0, F) = 3	Reduce 6 (F →a) GOTO (0, F) = 3	F + a * a $
0F3	+ a * a $	+	LR(3, +) = R4 LR(0, T) = 2	Reduce 4 (T → F) GOTO (0, T) = 2	T + a * a $
0T2	+ a * a $	+	LR(2, +) = R2 LR(0, E) = 1	Reduce 2 (E → T) GOTO (0, E) = 1	E + a * a $
0E1	+ a * a $	+	LR(1, +) = S6	Shift 6	E + a * a $
0E1+6	a * a $	a	LR(6, a) = S5	Shift 5	E + a * a $

0E1+6a5	* a $	*	LR(5, *) = R6 LR(6, F) = 3	Reduce 6 (F → a) GOTO (6, F) = 3	E + F * a $
0E1+6F3	* a $	*	LR(3, *) = R4 LR(6, T) = 9	Reduce 4 (T → F) GOTO (6, T) = 9	E + T * a $
0E1+6T9	* a $	*	LR(9, *) = S7	Shift 7	E + T * a $
0E1+6T9*7	a $	a	LR(7, a) = S5	Shift 5	E + T * a $
0E1+6T9*7a5	$	$	LR(5, $) = R6 LR(7, F) = 10	Reduce 6 (F → a) GOTO (7, F) = 10	E + T * F $
0E1+6T9*7F10	$	$	LR(10, $) =R3 LR(6, T) = 9	Reduce 3 (T → T*F) GOTO (6, T) = 9	E + T $
0E1+6T9	$	$	LR(9, $) = R1 LR(0, E) = 1	Reduce 1 (E → E + T) GOTO (0, E) = 1	E $
0E1	$	$	LR(1, $) = accept	Accept	

Reverse of Rightmost Derivation:

a + a * a

⇒ F + a * a

⇒ T + a * a

⇒ E + a * a

⇒ E + a * a

⇒ E + F * a

⇒ E + T * a

⇒ E + T * F

⇒ E + T

⇒ E