Bottom-Up Parsing - II

Lecture 8

Section 4.5, 4.6

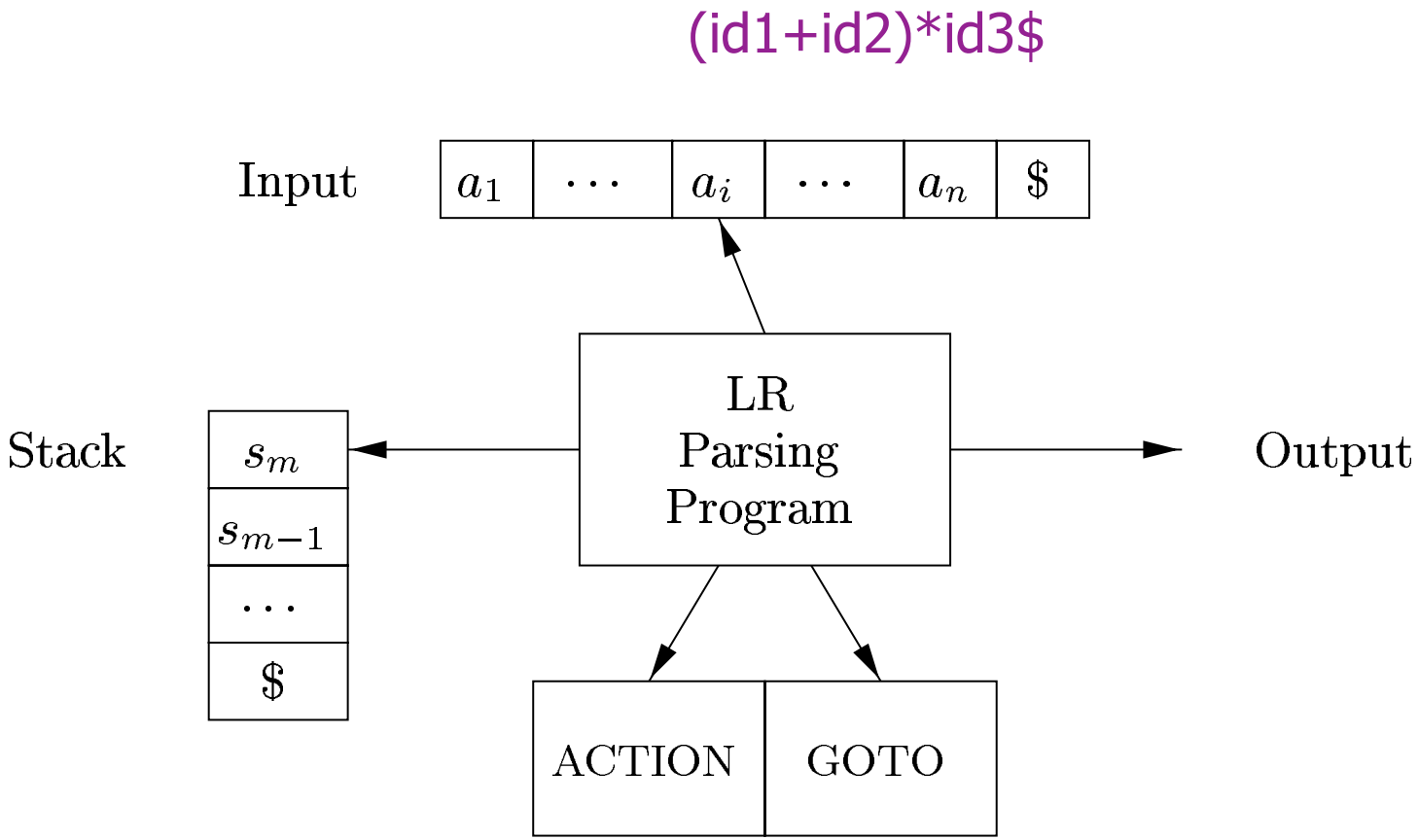
Rokan Uddin Faruqui

Associate Professor
Dept of Computer Science and Engineering
University of Chittaong, Bangladesh
Email: *rokan@cu.ac.bd*

1. LR Parsing
2. The Parsing Tables
3. The Action Table
4. The Goto Table
5. Assignment
6. The Algorithm

Outline

LR Parsing - Model



Outline

The LR(0) Parsing Tables

- There are two tables that we will construct.
- The action table contains shift and reduce actions to be taken upon processing terminals.
- The goto table contains changes of state upon matching productions.

Outline

The Action Table

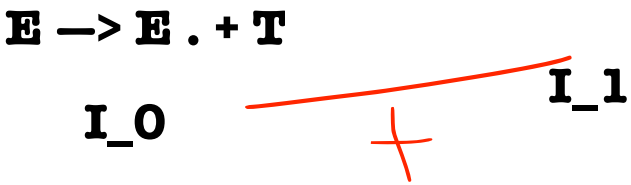
- The action table contains one row for each state in the PDA and one column for each terminal and EOF ($).
- The entries are
  - Shift n.
    - Push the current token and move to state n.
  - Reduce n.
    - Pop the symbols of the right side of production n and then push the nonterminal of the left side.
    - Then change state according to the goto table.

Building the Action Table

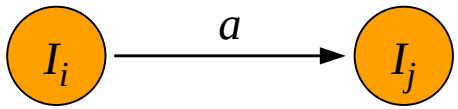
```
E —> E . + T
T—>T * F.
```

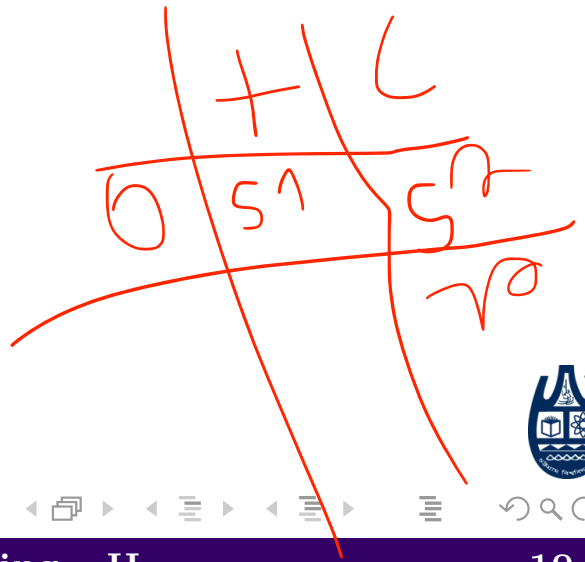
- The action table is built from items of the form $[A \rightarrow \alpha \bullet a\beta]$ and $[A \rightarrow \alpha\bullet]$.
  - Items $[A \rightarrow \alpha \bullet a\beta]$ produce shift operations.
  - Items $[A \rightarrow \alpha\bullet]$ produce reduce operations.
- The goto table is built from items of the form $[A \rightarrow \alpha \bullet B\beta]$.

Building the Action Table



- If $[A \rightarrow \alpha \bullet a\beta]$ is in state I_i and the PDA transition on a is from state I_i to state I_j, then set the (i, a) entry to "shift j."
- In the table, we write "sj," where j is the number of the destination state.
- Thus, every transition on a terminal becomes a shift entry in the action table.

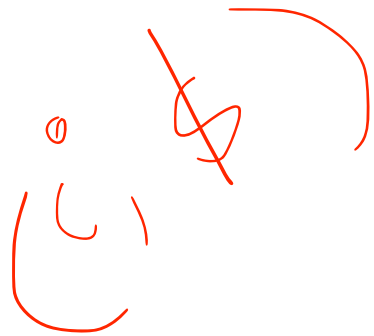




Building the Action Table

- If $[A \to \alpha\bullet] \in I_i$ and $[S' \to S\bullet] \notin I_i$, then set the (i, a) entry to "reduce $A \to \alpha$" for all a in FOLLOW(A).
- In the table, we write "rk," where k is the number of the production $A \to \alpha$.

Building the Action Table



- If $[S' \rightarrow S\bullet]$ is in state I_i, then the $(i, \$)$ entry is "accept."
- In the table, we write "acc."
- All empty cells are labeled "error."

Example

Example (FIRST and FOLLOW)

- To find the action table for our example, we need to know nullability, FIRST, and FOLLOW.

$$E' \to E$$
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid \mathbf{id}$$

Example

Example (FIRST and FOLLOW)

	Nullable	FIRST	FOLLOW
E'	No	$\{(, \mathbf{id}\}$	$\{\$\}$
E	No	$\{(, \mathbf{id}\}$	$\{\$, +,)\}$
T	No	$\{(, \mathbf{id}\}$	$\{\$, +,), *\}$
F	No	$\{(, \mathbf{id}\}$	$\{\$, +,), *\}$

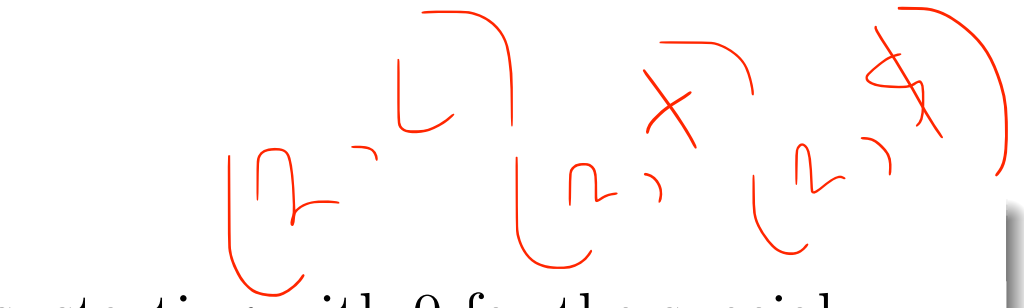
Example

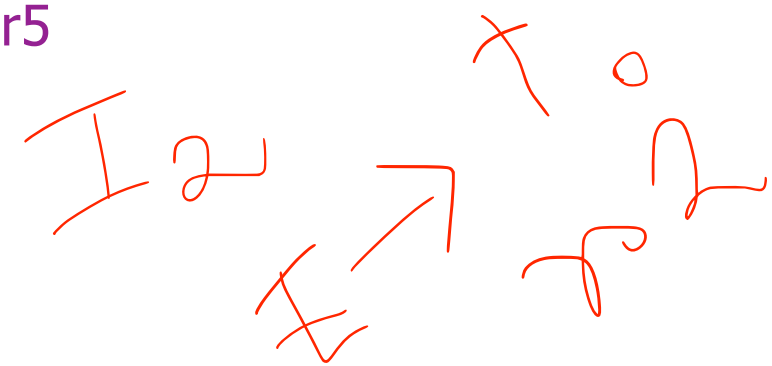
Example (The Action Table)

- Then number the productions, starting with 0 for the special production:

0. $E' \to E$
1. $E \to E + T$
2. $E \to T$
3. $T \to T * F$
4. $T \to E$
5. $F \to (E)$
6. $F \to \mathbf{id}$

s5
r5





Example

I_0 —–> I_4

Example (The Action Table)

I_0 —–> I_5

I2

E —>T.

R —>T.*F

	+	*	(	)	id	$
0			s4		s5	
1	s6					acc
2	r2	s7		r2		r2
3	r4	r4		r4		r4
4			s4		s5	
5	r6	r6		r6		r6
6			s4		s5	
7			s4		s5	
8	s6			s11		
9	r1	s7		r1		r1
10	r3	r3		r3		r3
11	r5	r5		r5		r5

Outline

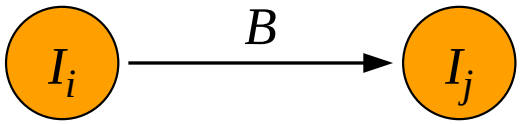
The Goto Table

- The goto table has one row for each state in the PDA and one column for each nonterminal except S'.
- The entries are states of the PDA.

Building the Goto Table

E —> E + T.

- If $[A \rightarrow \alpha \bullet B\beta]$ is in state I_i and the PDA transition is



  then set the (i, B) entry to "goto j."
- In the goto table, we write "j."
- Thus, every transition on a nonterminal becomes an entry in the goto table.

Example

Example (The Goto Table)

	E	T	F
0	1	2	3
1			
2			
3			
4	8	2	3
5			
6		9	3
7			10
8			
9			
10			
11			

Outline

Assignment

Homework

- Let the grammar be

$$S \rightarrow (\ L \ ) \mid \mathbf{id}$$
$$L \rightarrow L \ , \ S \mid S$$

- Build the action and goto tables.

Outline

INPUT: An input string w and an LR- parsing table with functions **ACTION and GOTO** for a grammar **G**

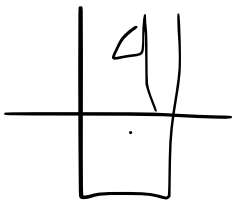
OUTPUT: If w is in $L(G)$ the reduction steps of a bottom- up parse for w otherwise an error indication.

METHOD: Initially the parser has s_0 on its stack, where s_0 is the initial state and w in the input buffer.

2. E—>E+T

let a be the first symbol of $w\$$;
while(1) { /* repeat forever */
    let s be the state on top of the stack;
    if (ACTION$[s, a]$ = shift t) {
        push t onto the stack;
        let a be the next input symbol;
    } **else if** (ACTION$[s, a]$ = reduce $A \rightarrow \beta$) {
        pop $|\beta|$ symbols off the stack;
        let state t now be on top of the stack;
        push GOTO$[t, A]$ onto the stack;
        output the production $A \rightarrow \beta$;
    } **else if** (ACTION$[s, a]$ = accept) break; /* parsing is done */
    else call error-recovery routine;
}

Action[4,(] = s4
Action[2,)] = r2
on[4,)] = blank (ERROR)
Action[2,$] = Accept



Example

Action[0,(] = s4

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 (4	id + id) ∗ id $
0 (4 id 5	+ id) ∗ id $
0 (4 F 3	+ id) ∗ id $
0 (4 T 2	+ id) ∗ id $
0 (4 E 8	+ id) ∗ id $
0 (4 E 8 + 6	id) ∗ id $
0 (4 E 8 + 6 id 5	) ∗ id $
0 (4 E 8 + 6 F 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Action[0,(] = s4
Action[4,id] = s5

Stack	Input
0	**(id** + id) ∗ id $
0 (4	**id** + id) ∗ id $
0 (4 id 5	+ id) ∗ id $
0 (4 F 3	+ id) ∗ id $
0 (4 T 2	+ id) ∗ id $
0 (4 E 8	+ id) ∗ id $
0 (4 E 8 + 6	id) ∗ id $
0 (4 E 8 + 6 id 5	) ∗ id $
0 (4 E 8 + 6 F 3	) ∗ id $

Example

0 (4 F

Example (Parse **(id + id)∗ id**)

Action[5,+] = r6

6. F —> id

Stack	Input
0	**(id + id) ∗ id $**
0 (4	**id + id) ∗ id $**
0 (4 **id** 5	**+ id) ∗ id $**
0 (4 *F* 3	+ id) ∗ id $
0 (4 *T* 2	+ id) ∗ id $
0 (4 *E* 8	+ id) ∗ id $
0 (4 *E* 8 + 6	id) ∗ id $
0 (4 *E* 8 + 6 id 5	) ∗ id $
0 (4 *E* 8 + 6 *F* 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Action[3,+]

Stack	Input
0	**(id + id) ∗ id $**
0 **(** 4	**id + id) ∗ id $**
0 **(** 4 **id** 5	**+ id) ∗ id $**
0 **(** 4 F 3	**+ id) ∗ id $**
0 **(** 4 T 2	+ id) ∗ id $
0 **(** 4 E 8	+ id) ∗ id $
0 **(** 4 E 8 + 6	id) ∗ id $
0 **(** 4 E 8 + 6 id 5	) ∗ id $
0 **(** 4 E 8 + 6 F 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 (4	**id + id) ∗ id $**
0 (4 **id** 5	**+ id) ∗ id $**
0 (4 F 3	**+ id) ∗ id $**
0 (4 $\underline{T}$ 2	**+ id) ∗ id $**
0 (4 E 8	+ id) ∗ id $
0 (4 E 8 + 6	id) ∗ id $
0 (4 E 8 + 6 id 5	) ∗ id $
0 (4 E 8 + 6 F 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id** $
0 **(** 4	**id + id) ∗ id** $
0 **(** 4 **id** 5	**+ id) ∗ id** $
0 **(** 4 F 3	**+ id) ∗ id** $
0 **(** 4 T 2	**+ id) ∗ id** $
0 **(** 4 E 8	**+ id) ∗ id** $
0 **(** 4 E 8 **+** 6	**id) ∗ id** $
0 **(** 4 E 8 **+** 6 **id** 5	**) ∗ id** $
0 **(** 4 E 8 **+** 6 F 3	**) ∗ id** $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 **(** 4	**id + id) ∗ id $**
0 **(** 4 **id** 5	**+ id) ∗ id $**
0 **(** 4 F 3	**+ id) ∗ id $**
0 **(** 4 T 2	**+ id) ∗ id $**
0 **(** 4 E 8	**+ id) ∗ id $**
0 **(** 4 E 8 **+** 6	**id) ∗ id $**
0 **(** 4 E 8 **+** 6 **id** 5	**) ∗ id $**
0 **(** 4 E 8 **+** 6 F 3	**) ∗ id $**

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 **(** 4	**id + id) ∗ id $**
0 **(** 4 **id** 5	**+ id) ∗ id $**
0 **(** 4 F 3	**+ id) ∗ id $**
0 **(** 4 T 2	**+ id) ∗ id $**
0 **(** 4 E 8	**+ id) ∗ id $**
0 **(** 4 E 8 **+** 6	**id) ∗ id $**
0 **(** 4 E 8 **+** 6 **id** 5	**) ∗ id $**
0 **(** 4 E 8 **+** 6 F 3	**) ∗ id $**

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 (4	**id + id) ∗ id $**
0 (4 **id** 5	**+ id) ∗ id $**
0 (4 F 3	**+ id) ∗ id $**
0 (4 T 2	**+ id) ∗ id $**
0 (4 E 8	**+ id) ∗ id $**
0 (4 E 8 + 6	**id) ∗ id $**
0 (4 E 8 + 6 **id** 5	**) ∗ id $**
0 (4 E 8 + 6 F 3	**) ∗ id $**

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ id \$
0 (4 E 8) 11	$*$ id \$
0 F 3	$*$ id \$
0 T 2	$*$ id \$
0 T 2 $*$ 7	id \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ id \$
0 F 3	$*$ id \$
0 T 2	$*$ id \$
0 T 2 $*$ 7	id \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ id \$
0 T 2	$*$ id \$
0 T 2 $*$ 7	id \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ id \$
0 T 2 $*$ 7	id \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

Example

Example (LR Parsing)

Stack	Input
0 (4 E 8 + 6 T 9	) $*$ **id** \$
0 (4 E 8	) $*$ **id** \$
0 (4 E 8) 11	$*$ **id** \$
0 F 3	$*$ **id** \$
0 T 2	$*$ **id** \$
0 T 2 $*$ 7	**id** \$
0 T 2 $*$ 7 **id** 5	\$
0 T 2 $*$ 7 F 10	\$
0 T 2	\$
0 E 1	\$
Accept	\$

- In practice, we need only push the state numbers onto the stack.
- There is no need to push the grammar symbols.
- Thus, the previous example can be streamlined.

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	id + id) ∗ id $
0 4 5	+ id) ∗ id $
0 4 3	+ id) ∗ id $
0 4 2	+ id) ∗ id $
0 4 8	+ id) ∗ id $
0 4 8 6	id) ∗ id $
0 4 8 6 5	) ∗ id $
0 4 8 6 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	+ id) ∗ id $
0 4 3	+ id) ∗ id $
0 4 2	+ id) ∗ id $
0 4 8	+ id) ∗ id $
0 4 8 6	id) ∗ id $
0 4 8 6 5	) ∗ id $
0 4 8 6 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	+ id) ∗ id $
0 4 2	+ id) ∗ id $
0 4 8	+ id) ∗ id $
0 4 8 6	id) ∗ id $
0 4 8 6 5	) ∗ id $
0 4 8 6 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	+ id) ∗ id $
0 4 8	+ id) ∗ id $
0 4 8 6	id) ∗ id $
0 4 8 6 5	) ∗ id $
0 4 8 6 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	**+ id) ∗ id $**
0 4 8	+ id) ∗ id $
0 4 8 6	id) ∗ id $
0 4 8 6 5	) ∗ id $
0 4 8 6 3	) ∗ id $

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	**+ id) ∗ id $**
0 4 8	**+ id) ∗ id $**
0 4 8 6	**id) ∗ id $**
0 4 8 6 5	**) ∗ id $**
0 4 8 6 3	**) ∗ id $**

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	**+ id) ∗ id $**
0 4 8	**+ id) ∗ id $**
0 4 8 6	**id) ∗ id $**
0 4 8 6 5	**) ∗ id $**
0 4 8 6 3	**) ∗ id $**

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	**+ id) ∗ id $**
0 4 8	**+ id) ∗ id $**
0 4 8 6	**id) ∗ id $**
0 4 8 6 5	**) ∗ id $**
0 4 8 6 3	**) ∗ id $**

Example

Example (Parse **(id + id)∗ id**)

Stack	Input
0	**(id + id) ∗ id $**
0 4	**id + id) ∗ id $**
0 4 5	**+ id) ∗ id $**
0 4 3	**+ id) ∗ id $**
0 4 2	**+ id) ∗ id $**
0 4 8	**+ id) ∗ id $**
0 4 8 6	**id) ∗ id $**
0 4 8 6 5	**) ∗ id $**
0 4 8 6 3	**) ∗ id $**

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) ∗ **id** $
0 4 8	) ∗ id $
0 4 8 11	∗ id $
0 3	∗ id $
0 2	∗ id $
0 2 7	id $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* id $
0 3	* id $
0 2	* id $
0 2 7	id $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* id $
0 2	* id $
0 2 7	id $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* id $
0 2 7	id $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (LR Parsing)

Stack	Input
0 4 8 6 9	) * **id** $
0 4 8	) * **id** $
0 4 8 11	* **id** $
0 3	* **id** $
0 2	* **id** $
0 2 7	**id** $
0 2 7 5	$
0 2 7 10	$
0 2	$
0 1	$
Accept	$

Example

Example (A Simplified Grammar)

- We may simplify our grammar to

$$E \to E + E$$
$$E \to E * E$$
$$E \to (E)$$
$$E \to \mathbf{id}$$

- In this form, the precedence rules for + and * are not implicit.
- They must be incorporated into the tables.

Assignment

Homework

- The grammar for if and if-else statements is

$$
\begin{aligned}
S \rightarrow\ & \texttt{if (} E \texttt{)} \ S \\
& \mid \texttt{if (} E \texttt{)} \ S \ \texttt{else} \ S \\
& \mid \textbf{id = num} \\
E \rightarrow\ & \textbf{id == num}
\end{aligned}
$$

  where S is a statement and E is an expression (boolean).

- Write the action and goto tables for this grammar.
- Find the shift/reduce conflict(s) and decide how to handle them.

(continued...)