

Compiler Construction Course

Lecture 6

Syntax Analysis

Bottom-Up Parsing (Part 2)

LR(0) Parsers

Sayyed Kazem Shekofteh

LR(0) Parsing Table

- Generating an LR parsing table consists identifying the possible states and arranging the transitions among them.
- Definition of **Configuration** or **item**
 - *A configuration is a production of the grammar with a dot at some position on its right side.*
- For example $A \rightarrow XYZ$ has four possible items:
 - $A \rightarrow \bullet XYZ$
 - $A \rightarrow X \bullet YZ$
 - $A \rightarrow XY \bullet Z$
 - $A \rightarrow XYZ \bullet$
- This dot marks how far we have gotten in parsing the production.
 - A dot in the middle
 - A dot at the right end

LR(0) Parsing Table

- For example, if we are currently in this position:

$$A \rightarrow X \bullet YZ$$

- Naturally We want to shift something from $\text{First}(Y)$.

- Say we have productions $Y \rightarrow u \mid w$

- Given that, these three productions all correspond to the same state of the shift-reduce parser:

$$A \rightarrow X \bullet YZ$$

$$Y \rightarrow \bullet u$$

$$Y \rightarrow \bullet w$$

- At the above point in parsing, we have just recognized an X and expect the upcoming input to contain a sequence derivable from YZ .

- the sequence to be derivable from either u or w .

LR(0) Parsing Table

$A \rightarrow X \bullet YZ$

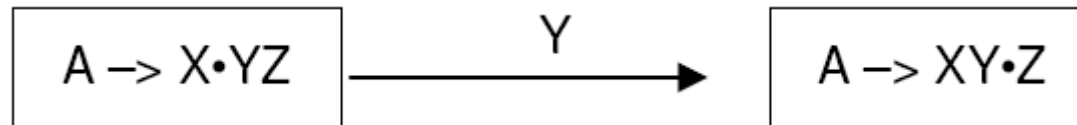
$Y \rightarrow \bullet u$

$T \rightarrow \bullet w$

- We can put these three items into a set and call it a **configuration set** of the LR parser.
 - **Closure:** The action of adding equivalent configurations to create a configuration set.
 - Our parsing tables will have one state corresponding to each configuration set.
- ① The states can be mapped to states of a DFA and also their construction step.

LR(0) Parsing Table

- We move from one state to another via transitions marked with a symbol of the CFG.



- We shift until we reach a state where the dot is at the end of a production, at which point we reduce.
- Now for the formal rule for what to put in a configuration set, We start with a configuration:

$$A \rightarrow X_1 \dots X_i \cdot X_{i+1} \dots X_j$$

- We then perform the closure operation. So, if we have

$$X_{i+1} \rightarrow Y_1 \dots Y_g \mid Z_1 \dots Z_h$$

- We would add the following to the configuration set.

$$X_{i+1} \rightarrow \cdot Y_1 \dots Y_g$$

$$X_{i+1} \rightarrow \cdot Z_1 \dots Z_h$$

LR(0) Parsing Table

- Repeat until when?
- We repeat this operation for all configurations in the configuration set where a dot precedes a nonterminal until no more configurations can be added.
- General instruction to create a configuration set for the starting configuration $A \rightarrow \bullet u$, we follow the closure operation:
 1. $A \rightarrow \bullet u$ is in the configuration set
 2. If u begins with a terminal, we are done with this production.
 3. If u begins with a nonterminal B , add all productions with B on the left side, with the dot at the start of the right side: $B \rightarrow \bullet v$
 4. Repeat steps 2 and 3 for any productions added in step 3. Continue until you reach a fixed point.

LR(0) Parsing Table

- The other information we need to build our tables is the transitions between configuration sets.
- Definition of **Successor** function.
- Given a configuration set C and a grammar symbol X , the successor function computes the successor configuration set $C' = \text{successor}(C, X)$.
- The successor function: describes what set the parser moves to upon recognizing a given symbol.

LR(0) Parsing Table

- Computing the successor function:
 - We take all the configurations in C where there is a dot preceding X
 - Move the dot past X
 - Put the new configurations in C'
 - Apply the closure operation to C'
- An example:

$$E \rightarrow E \cdot + T$$

- To obtain the successor configuration set on $+$
- First put the following configuration in C' :

$$E \rightarrow E + \cdot T$$

- Then perform a closure on this set:

$$E \rightarrow E + \cdot T$$

$$T \rightarrow \cdot (E)$$

$$T \rightarrow \cdot id$$

LR(0) Parsing Table

- Creating the **action** and **goto** tables:
 - We need to construct all the configuration sets and successor functions for the expression grammar.
 - Why?
 - *Augmented Grammar*
 - We add $S' \rightarrow S$ to grammar.
- We start with the initial configuration set C_0 which is the closure of $S' \rightarrow \cdot S$

LR(0) Parsing Table

- The augmented grammar for the example expression grammar:

$E' \rightarrow E$

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow (E)$

$T \rightarrow id$

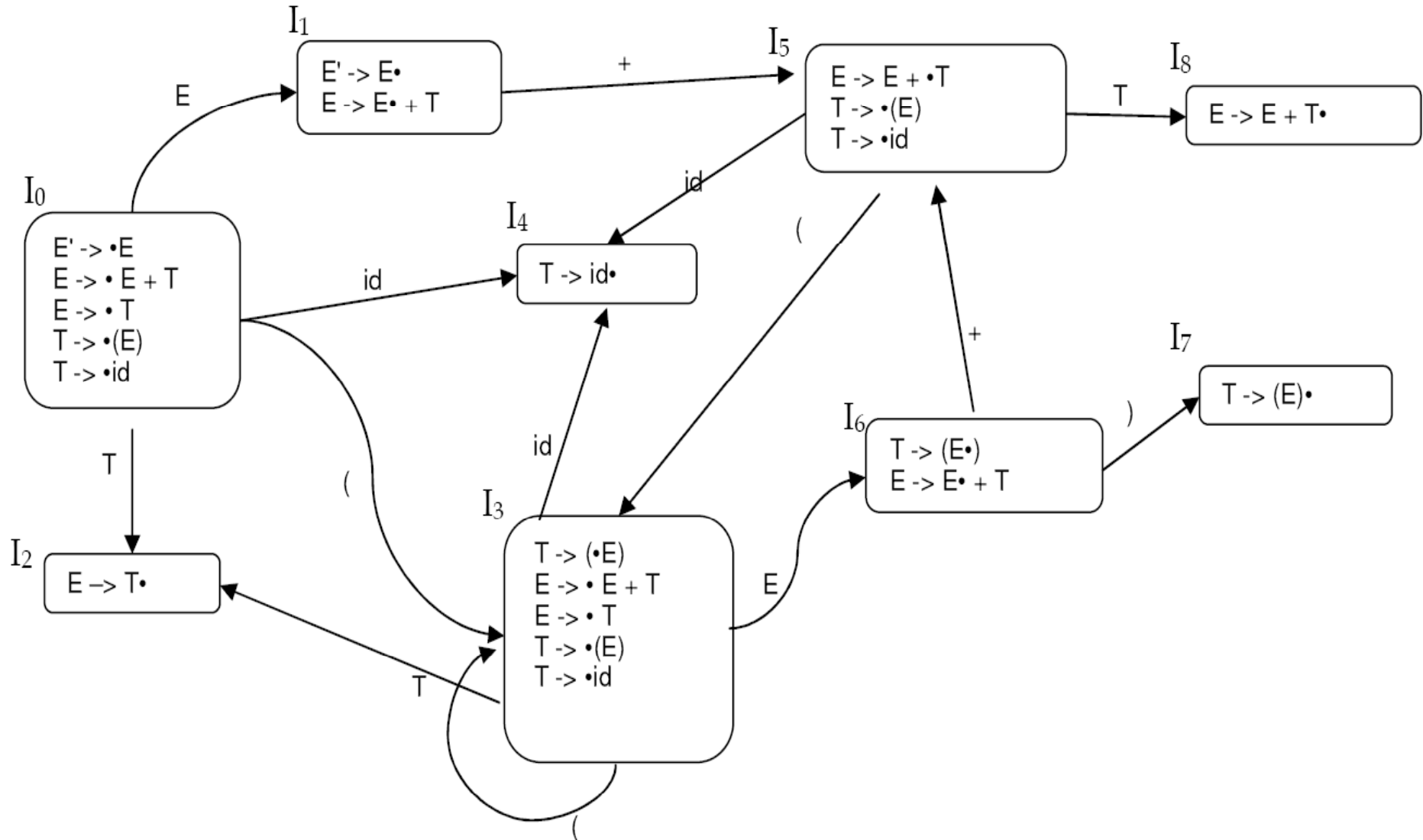
LR(0) Parsing Table

Configuration Set	Successor	Configuration Set	Successor
$I_0:$	$E' \rightarrow \bullet E$ $E \rightarrow \bullet E + T$ $E \rightarrow \bullet T$ $T \rightarrow \bullet (E)$ $T \rightarrow \bullet id$	I_1 I_1 I_2 I_3 I_4	$I_4:$ $T \rightarrow id \bullet$ Reduce 4 $I_5:$ $E \rightarrow E + \bullet T$ I_8 $T \rightarrow \bullet (E)$ I_3 $T \rightarrow \bullet id$ I_4
$I_1:$	$E' \rightarrow E \bullet$ Accept $E \rightarrow E \bullet + T$ I_5		$I_6:$ $T \rightarrow (E \bullet)$ I_7 $E \rightarrow E \bullet + T$ I_5
$I_2:$	$E \rightarrow T \bullet$ Reduce 2		
$I_3:$	$T \rightarrow (\bullet E)$ I_6 $E \rightarrow \bullet E + T$ I_6 $E \rightarrow \bullet T$ I_2 $T \rightarrow \bullet (E)$ I_3 $T \rightarrow \bullet id$ I_4		$I_7:$ $T \rightarrow (E) \bullet$ Reduce 3 $I_8:$ $E \rightarrow E + T \bullet$ Reduce 1

LR(0) Parsing Table

- Is numbering the sets important?
- To better visualize the configuration sets and successors, we can use a *goto-graph* or *transition diagram*.

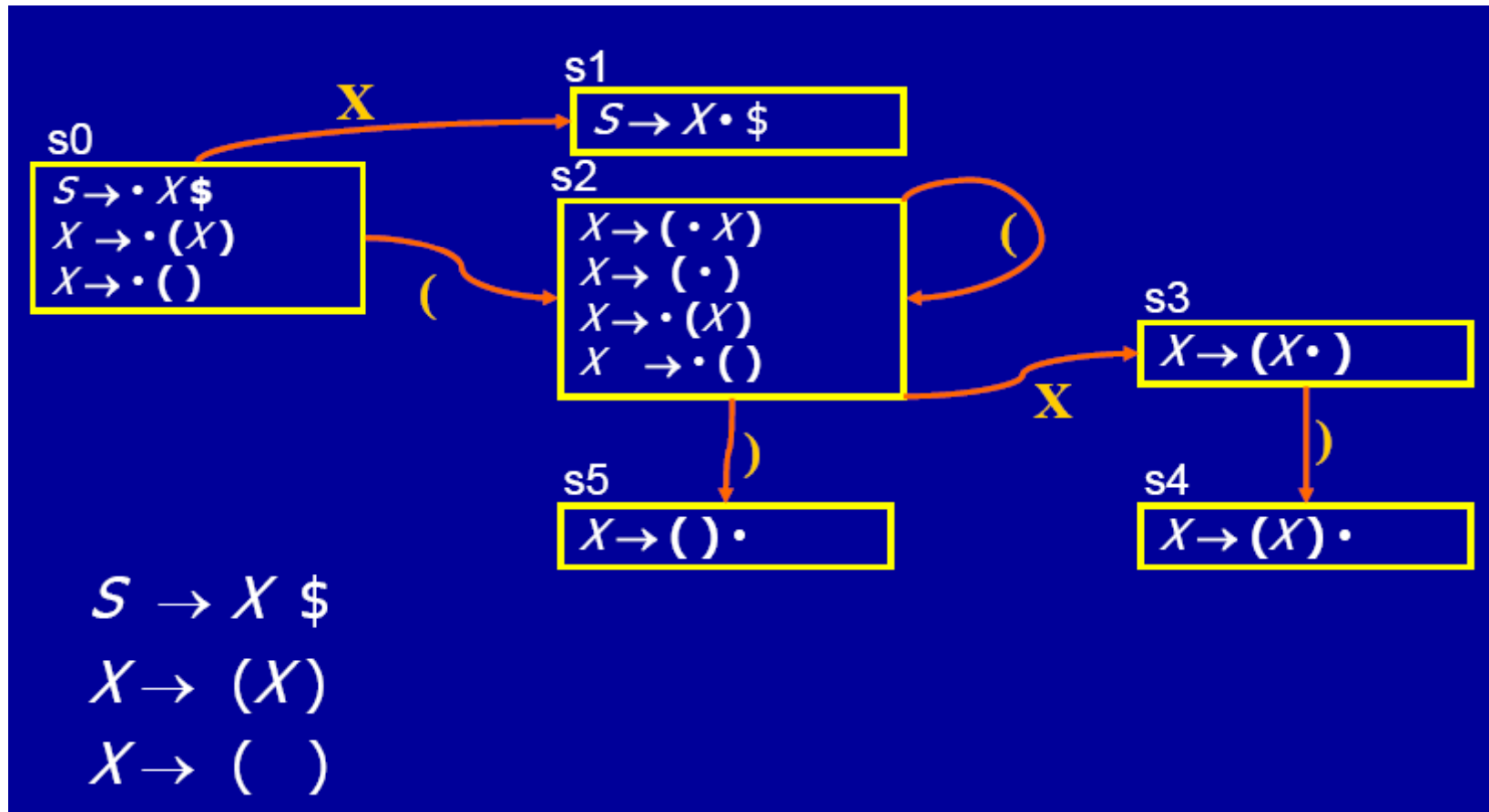
LR(0) Parsing Table



LR(0) Parsing Table

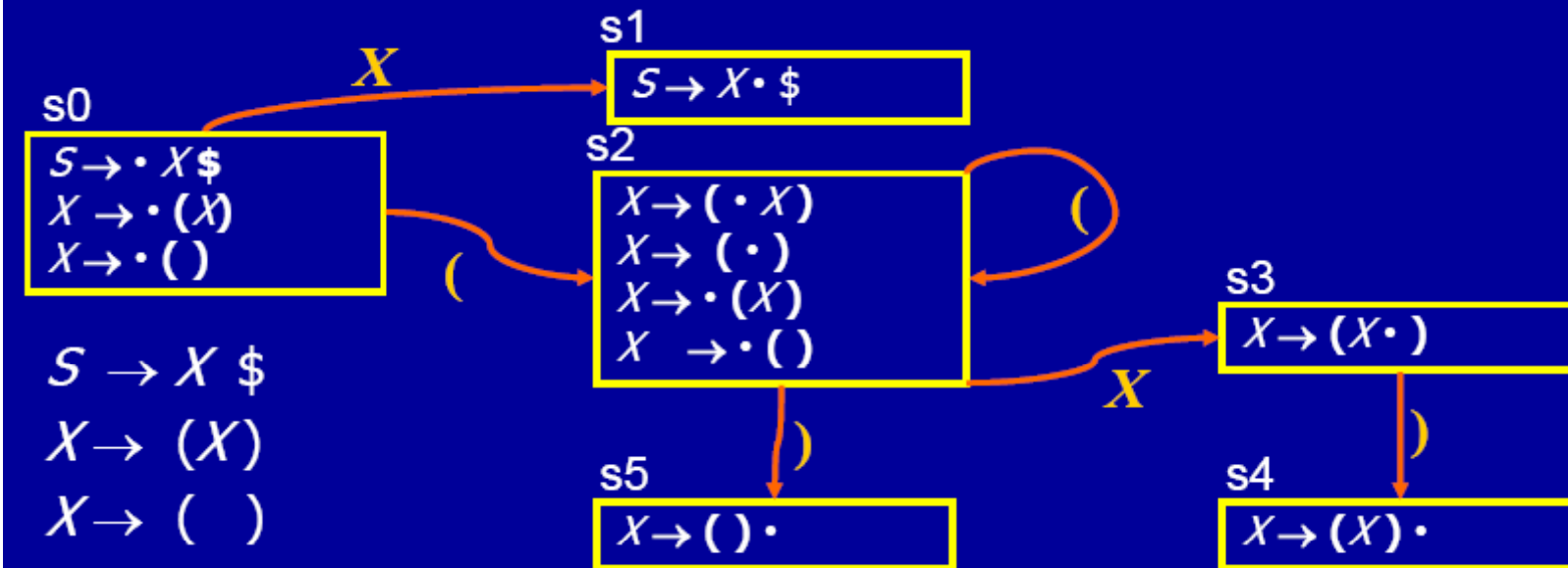
- To construct the LR(0) table using the DFA:
 1. Construct $\{I_0, I_1, \dots, I_n\}$, the collection of configuration sets for G' .
 2. State i is determined from I_i . The parsing actions for the state are determined as follows:
 - a) If $A \rightarrow u\bullet$ is in I_i then set $\text{Action}[i,a]$ to reduce $A \rightarrow u$ for all input. (A not equal to S').
 - b) If $S' \rightarrow S\bullet$ is in I_i then set $\text{Action}[i,\$]$ to accept.
 - c) If $A \rightarrow u\bullet av$ is in I_i and $\text{successor}(I_i, a) = I_j$, then set $\text{Action}[i,a]$ to shift j (a is a terminal)
 3. The goto transitions for state i are constructed for all nonterminals A using the rule: If $\text{successor}(I_i, A) = I_j$, then $\text{Goto}[i,A] = j$.
 4. All entries not defined by rules 2 and 3 are errors.
 5. The initial state is the one constructed from the configuration set containing $S' \rightarrow \bullet S$.

LR(0) Parsing Table - Example



LR(0) Parsing Table - Example

State	ACTION				Goto
	()	\$	X	
s0	shift to s2	error	error	goto s1	
s1	error	error	accept		
s2	shift to s2	shift to s5	error	goto s3	
s3	error	shift to s4	error		
s4	reduce (2)	reduce (2)	reduce (2)		
s5	reduce (3)	reduce (3)	reduce (3)		



LR(0) Parsing Table

- To construct the LR(0) table using the DFA:
 1. Construct $\{I_0, I_1, \dots, I_n\}$, the collection of configuration sets for G' .
 2. State i is determined from I_i . The parsing actions for the state are determined as follows:
 - a) If $A \rightarrow u\bullet$ is in I_i then set $\text{Action}[i,a]$ to reduce $A \rightarrow u$ for all input. (A not equal to S').
 - b) If $S' \rightarrow S\bullet$ is in I_i then set $\text{Action}[i,\$]$ to accept.
 - c) If $A \rightarrow u\bullet av$ is in I_i and $\text{successor}(I_i, a) = I_j$, then set $\text{Action}[i,a]$ to shift j (a is a terminal)
 3. The goto transitions for state i are constructed for all nonterminals A using the rule: If $\text{successor}(I_i, A) = I_j$, then $\text{Goto}[i,A] = j$.
 4. All entries not defined by rules 2 and 3 are errors.
 5. The initial state is the one constructed from the configuration set containing $S' \rightarrow \bullet S$.

LR(0) Parsing Table

- To be precise, a grammar is LR(0) if the following two conditions hold:
 1. For any configuration set containing the item $A \rightarrow u \bullet x v$ there is no complete item $B \rightarrow w \bullet$ in that set.
 - Translates to no shift-reduce conflict on any state.
 2. There is at most one complete item $A \rightarrow u \bullet$ in each configuration set.
 - Translates to no reduce-reduce conflict on any state.
- Very few grammars meet the requirements to be LR(0).
- For example, any grammar with an ϵ -rule will be problematic.

LR(0) Parsing Table

- Now consider the following grammar

$$S \rightarrow X \$$$
$$X \rightarrow a$$
$$X \rightarrow a b$$

LR(0) Parsing Table

	ACTION			Goto
State	a	b	\$	X
s0	shift to s1	error	error	goto s3
s1	reduce(2)	S/R Conflict	reduce(2)	
s2	reduce(3)	reduce(3)	reduce(3)	
s3	error	error	accept	

