

Compiler Design

Fatemeh Deldar

Isfahan University of Technology

1403-1404

Introduction to LR Parsing

- The most prevalent type of bottom-up parser: LR(k)
 - "L" is for left-to-right scanning of the input
 - "R" is for constructing a rightmost derivation in reverse
 - k is for the number of input symbols of lookahead that are used in making parsing decisions
- LR parsers are table-driven, much like the non-recursive LL parsers

Items and the LR(0) Automaton

- An LR parser makes shift-reduce decisions by maintaining states to keep track of where we are in a parse
- **States** represent sets of **items**
- **An LR(0) item of a grammar G** is a production of G with a dot at some position of the body
- **Example:** Production $A \rightarrow XYZ$ yields the four items:

$$\begin{aligned}A &\rightarrow \cdot XYZ \\A &\rightarrow X \cdot YZ \\A &\rightarrow XY \cdot Z \\A &\rightarrow XYZ \cdot\end{aligned}$$

- The production $A \rightarrow \varepsilon$ generates only one item, $A \rightarrow \cdot$.

Items and the LR(0) Automaton

- **Canonical LR(0) collection** provides the basis for constructing an LR(0) automaton
- To construct the canonical LR(0) collection for a grammar, we define an augmented grammar and two functions, **CLOSURE** and **GOTO**
- *If G is a grammar with start symbol S , then the augmented grammar for G , is G with a new start symbol S_0 and production $S_0 \rightarrow S$*
- **Closure of Item Sets**
 - If I is a set of items for a grammar G , then **CLOSURE(I)** is the set of items constructed from I by the two rules:
 1. Initially, add every item in I to **CLOSURE(I)**.
 2. If $A \rightarrow \alpha \cdot B\beta$ is in **CLOSURE(I)** and $B \rightarrow \gamma$ is a production, then add the item $B \rightarrow \cdot\gamma$ to **CLOSURE(I)**, if it is not already there. Apply this rule until no more new items can be added to **CLOSURE(I)**.

Items and the LR(0) Automaton

- **Example:** Consider the augmented expression grammar:

$$\begin{array}{lcl} E' & \rightarrow & E \\ E & \rightarrow & E + T \quad | \quad T \\ T & \rightarrow & T * F \quad | \quad F \\ F & \rightarrow & (E) \quad | \quad \text{id} \end{array}$$

- If I is the set of one item $\{[E' \rightarrow .E]\}$, then CLOSURE(I) is:

$E' \rightarrow \cdot E$
$E \rightarrow \cdot E + T$
$E \rightarrow \cdot T$
$T \rightarrow \cdot T * F$
$T \rightarrow \cdot F$
$F \rightarrow \cdot (E)$
$F \rightarrow \cdot \text{id}$

Items and the LR(0) Automaton

- We divide all the sets of items of interest into two classes:
 - **Kernel items:** the initial item, $S' \rightarrow .S$, and all items whose dots are not at the left end
 - **Non-kernel items:** all items with their dots at the left end, except for $S' \rightarrow .S$

$E' \rightarrow \cdot E$	Kernel items
$E \rightarrow \cdot E + T$	
$E \rightarrow \cdot T$	
$T \rightarrow \cdot T^* F$	
$T \rightarrow \cdot F$	
$F \rightarrow \cdot (E)$	
$F \rightarrow \cdot id$	

Non-kernel items

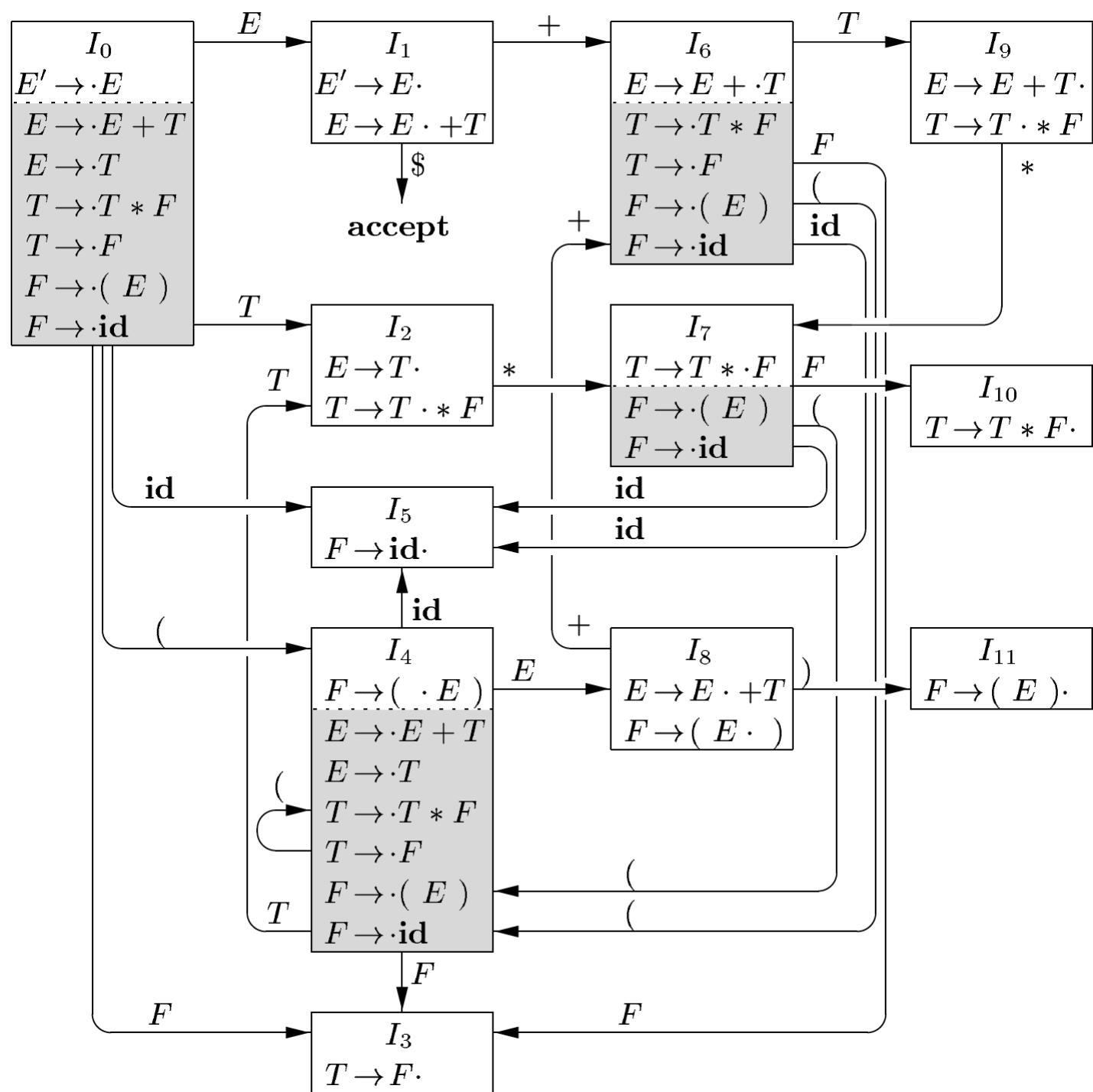
Items and the LR(0) Automaton

- **The Function GOTO**

- The GOTO function is used to define the transitions in the LR(0) automaton for a grammar
- **Example:** If I is the set of two items $\{[E' \rightarrow E.]$, $[E \rightarrow E. + T]\}$, then $GOTO(I, +)$ contains the items

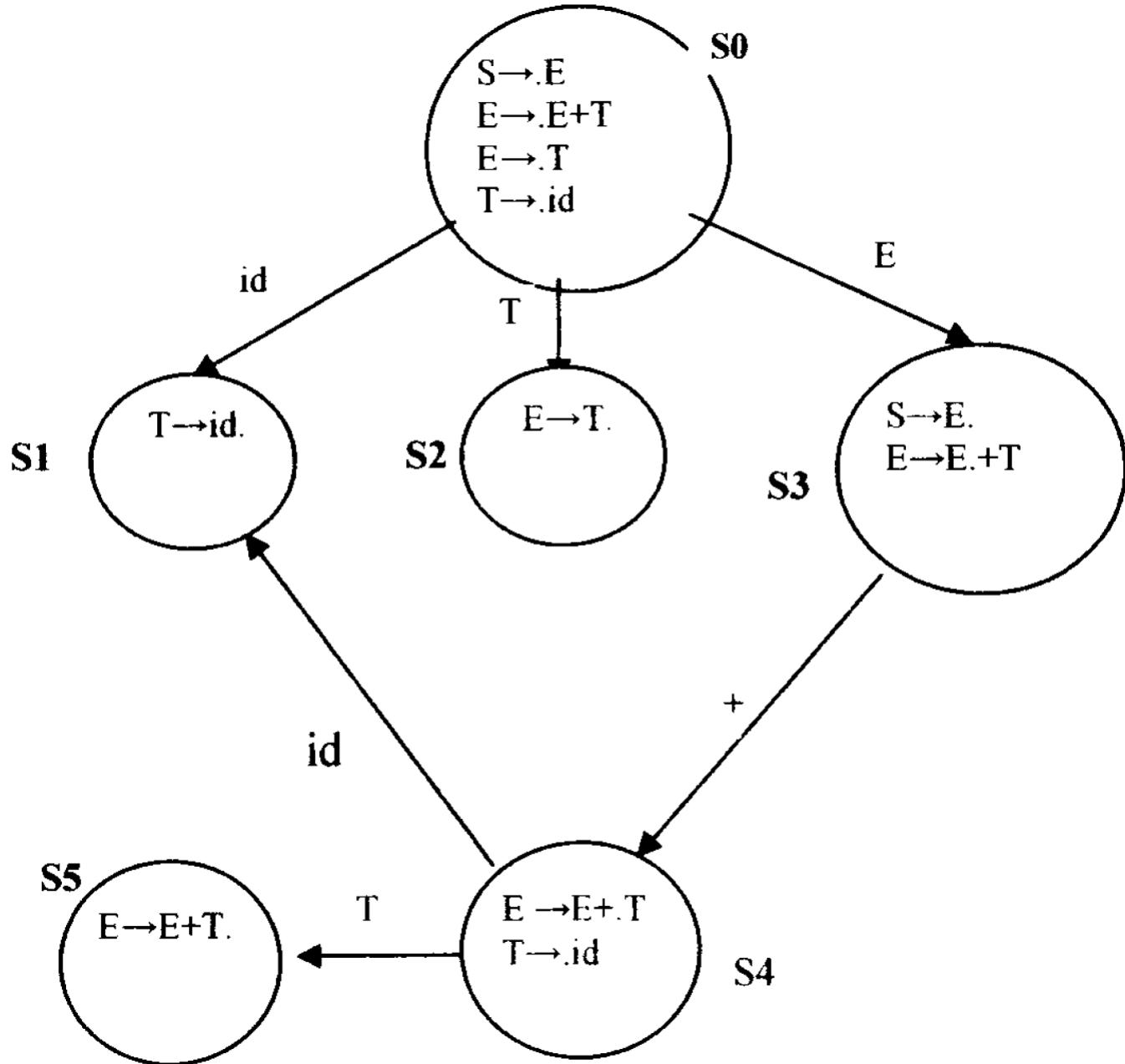
$$\begin{aligned}E &\rightarrow E + \cdot T \\T &\rightarrow \cdot T * F \\T &\rightarrow \cdot F \\F &\rightarrow \cdot(E) \\F &\rightarrow \cdot\text{id}\end{aligned}$$

- **Example: The canonical collection and GOTO function**



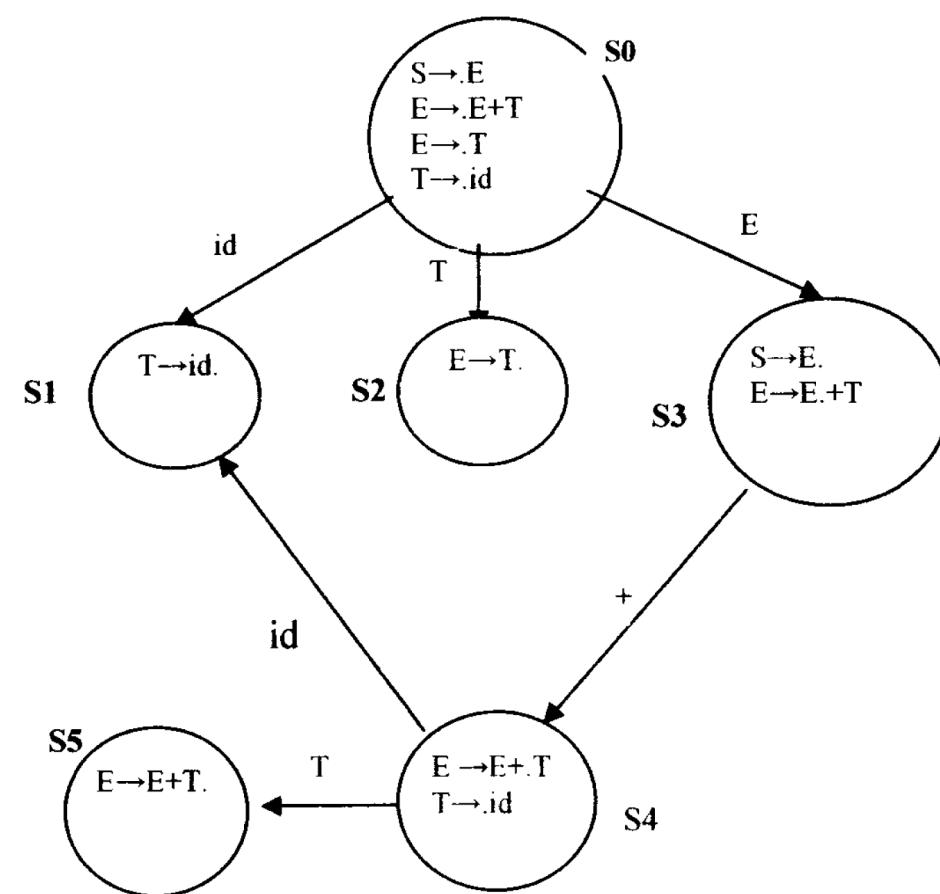
- Example:

- 1- $S \rightarrow E$
- 2- $E \rightarrow E + T$
- 3- $E \rightarrow T$
- 4- $T \rightarrow id$



LR(0) Parse Table

- The reduction is performed for all terminals
- If there are collisions in the LR(0) parse table cells, the grammar is not LR(0)*

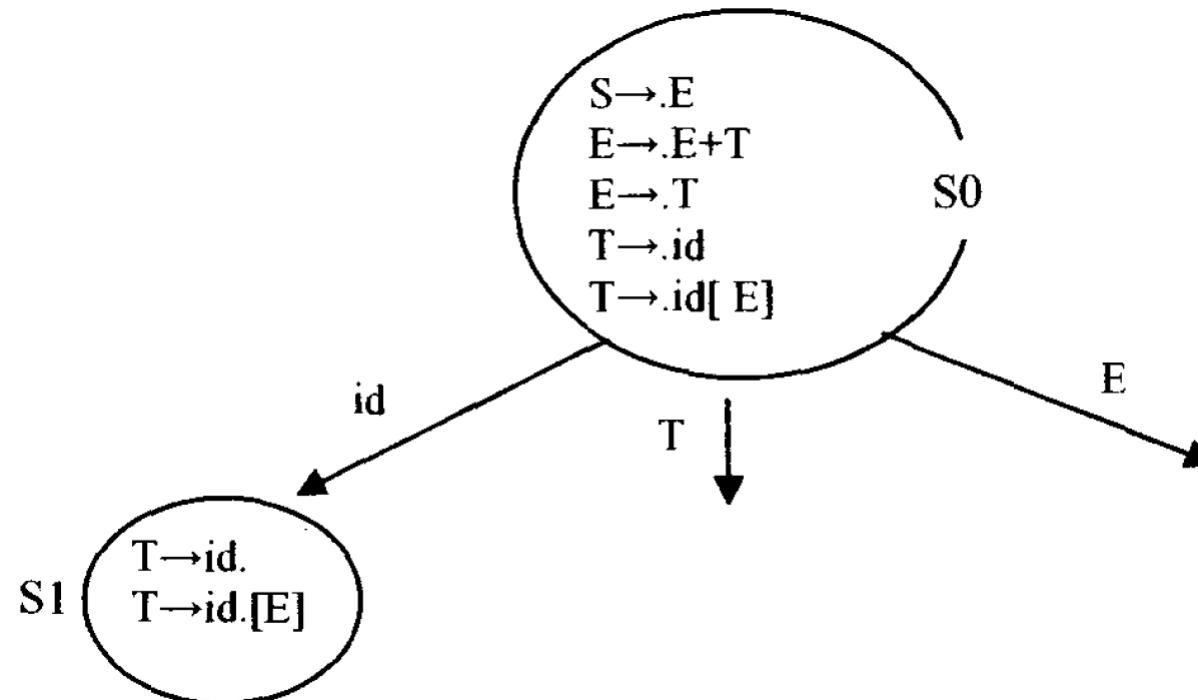


حالات	action				goto	
	id	+	\$	T	E	
0	s1	error	error	2	3	
1	r4	r4	r4			
2	r3	r3	r3			
3	error	s4	accept			
4	s1	error	error	5		
5	r2	r2	r2			

LR(0) Grammar

- Example: Shift/Reduce Conflict
 - The grammar is not LR(0)

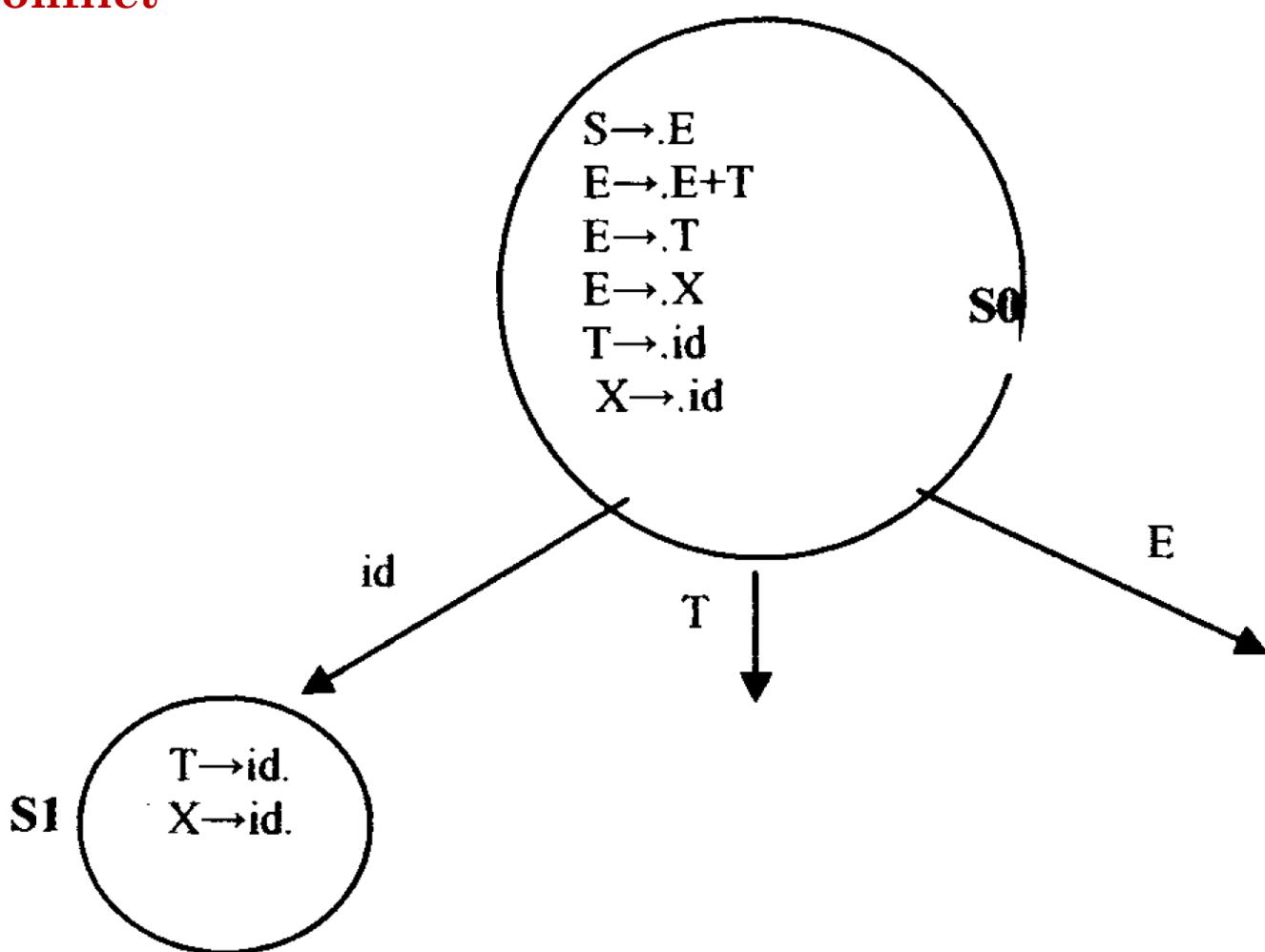
$S \rightarrow E$
 $E \rightarrow E + T$
 $E \rightarrow T$
 $T \rightarrow id$
 $T \rightarrow id [E]$



LR(0) Grammar

- Example: Reduce/Reduce Conflict
 - The grammar is not LR(0)

$S \rightarrow E$
 $E \rightarrow E + T$
 $E \rightarrow T$
 $E \rightarrow X$
 $T \rightarrow id$
 $X \rightarrow id$



LR(0) Grammar

- Example: Shift/Reduce Conflict
 - The grammar is not LR(0)

$S \rightarrow E$
 $E \rightarrow E + T$
 $E \rightarrow T$
 $T \rightarrow \epsilon$
 $T \rightarrow id$

