Compiler Design

Fatemeh Deldar

Isfahan University of Technology

1402-1403

Bottom-Up Parsing

- **Example:** Grammer $S \rightarrow 0\ S\ 1 \mid 0\ 1$ Input 000111

Stack	Input	Handle	Action
$	000111$		Shift
$0	00111$		Shift
$00	0111$		Shift
$000	111$		Shift
$0001	11$	01	Reduce: S -> 01
$00S	11$		Shift
$00S1	1$	0S1	Reduce : S -> 0S1
$0S	1$		Shift
$0S1	$	0S1	Reduce : S -> 0S1
$S	$		Accept

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 → XYZ yields the four items:

$$\begin{array}{l} A \rightarrow \cdot XYZ \\ A \rightarrow X{\cdot}YZ \\ A \rightarrow XY{\cdot}Z \\ A \rightarrow XYZ\cdot \end{array}$$

- The production A → ε generates only one item, A → .

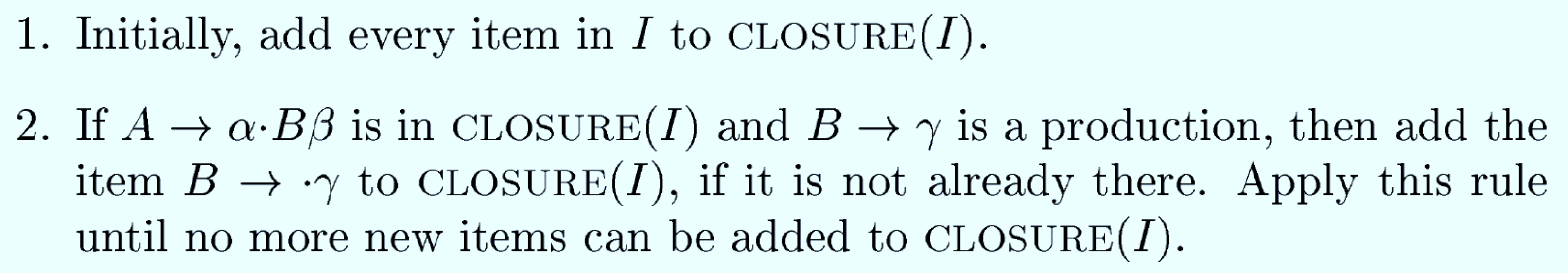
Items and the LR(0) Automaton

- **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 S0 and production S0 → 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 $\textsc{closure}(I)$.

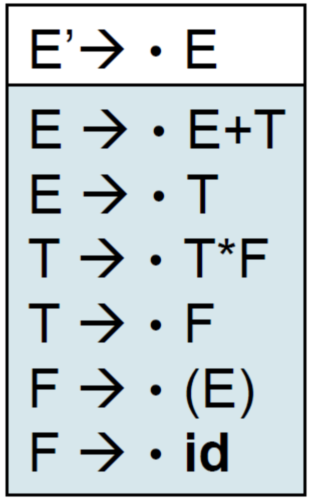
2. If $A \to \alpha{\cdot}B\beta$ is in $\textsc{closure}(I)$ and $B \to \gamma$ is a production, then add the item $B \to {\cdot}\gamma$ to $\textsc{closure}(I)$, if it is not already there. Apply this rule until no more new items can be added to $\textsc{closure}(I)$.

Items and the LR(0) Automaton

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

$$\begin{array}{rcl} E' & \to & E \\ E & \to & E + T \ \mid \ T \\ T & \to & T * F \ \mid \ F \\ F & \to & (E) \ \mid \ \mathbf{id} \end{array}$$

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



$E' \to \cdot E$

$E \to \cdot E+T$

$E \to \cdot T$

$T \to \cdot T*F$

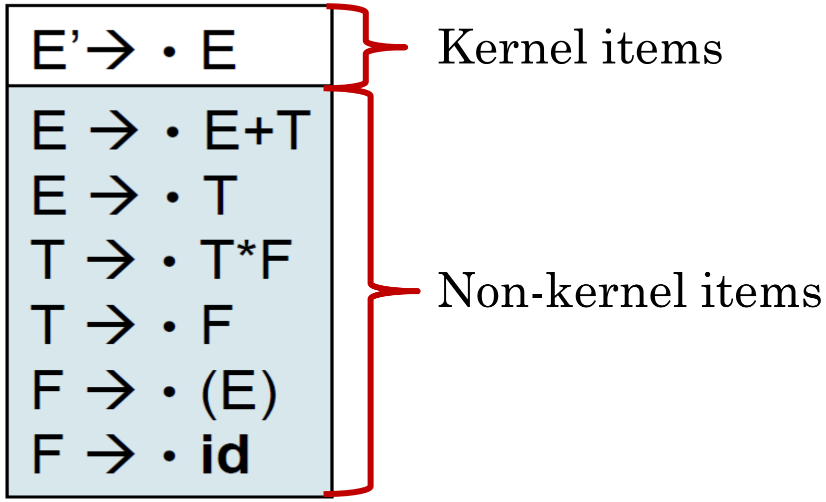
$T \to \cdot F$

$F \to \cdot (E)$

$F \to \cdot \mathbf{id}$

Items and the LR(0) Automaton

- We divide all the sets of items of interest into two classes:
  - **Kernel items:** the initial item, S' → .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' → .S



E’→ · E

E → · E+T

E → · T

T → · T*F

T → · F

F → · (E)

F → · **id**

Kernel items

Non-kernel items

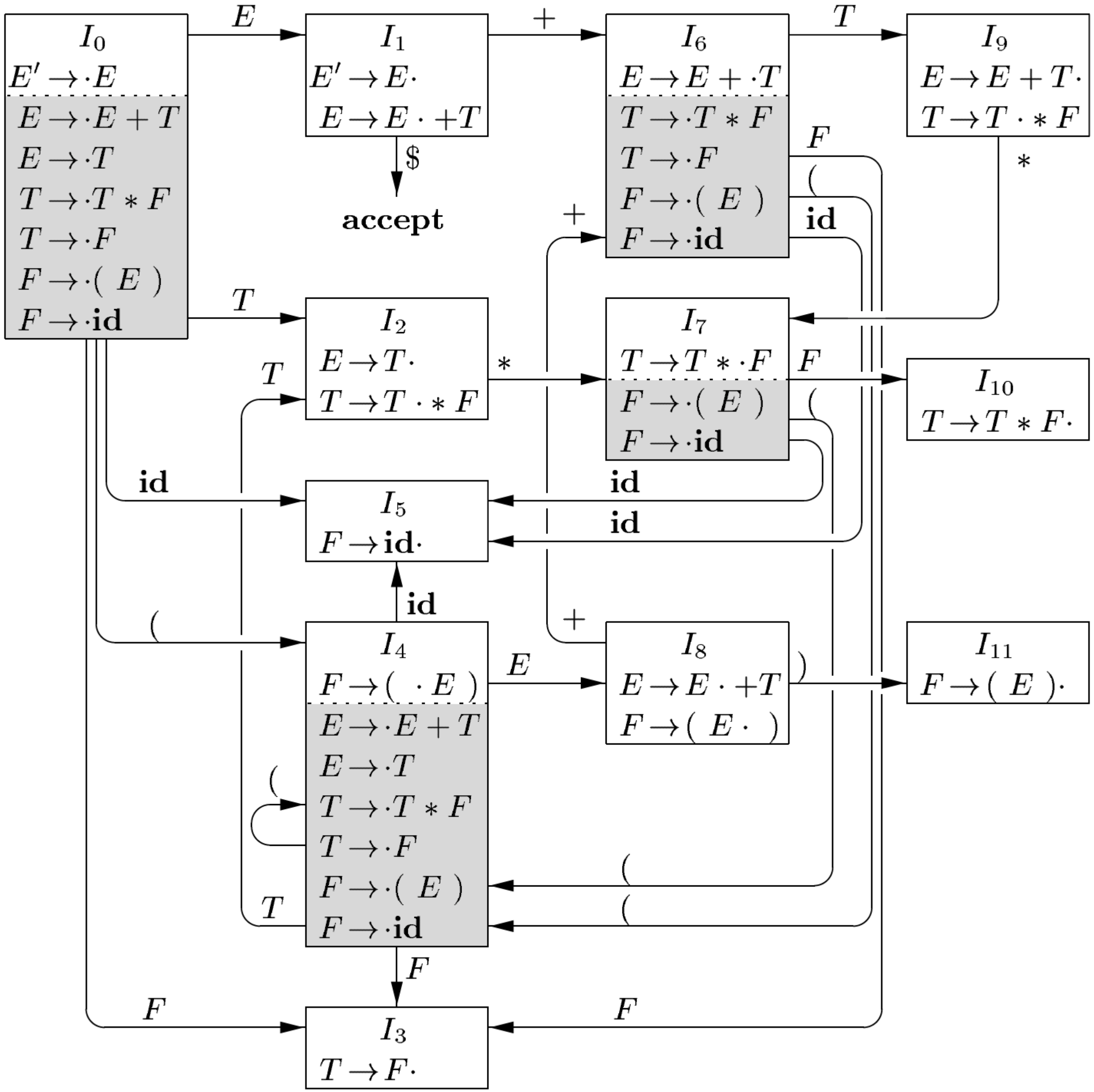
Items and the LR(0) Automaton

- **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' → E.], [E → E. + T]}, then GOTO(I, +) contains the items**

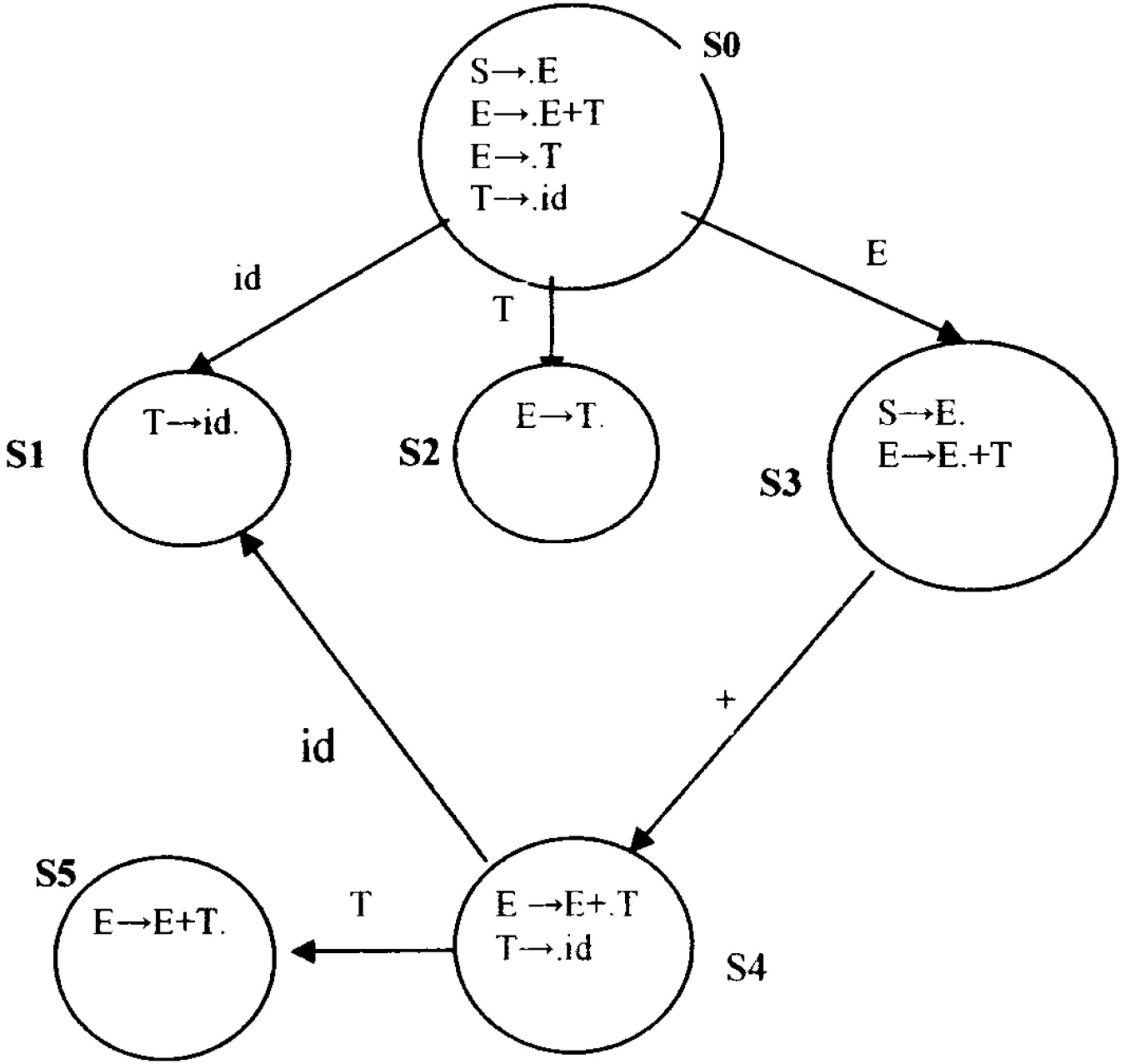
$$\begin{aligned} &E \to E + \cdot T \\ &T \to \cdot T * F \\ &T \to \cdot F \\ &F \to \cdot (E) \\ &F \to \cdot \mathbf{id} \end{aligned}$$

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



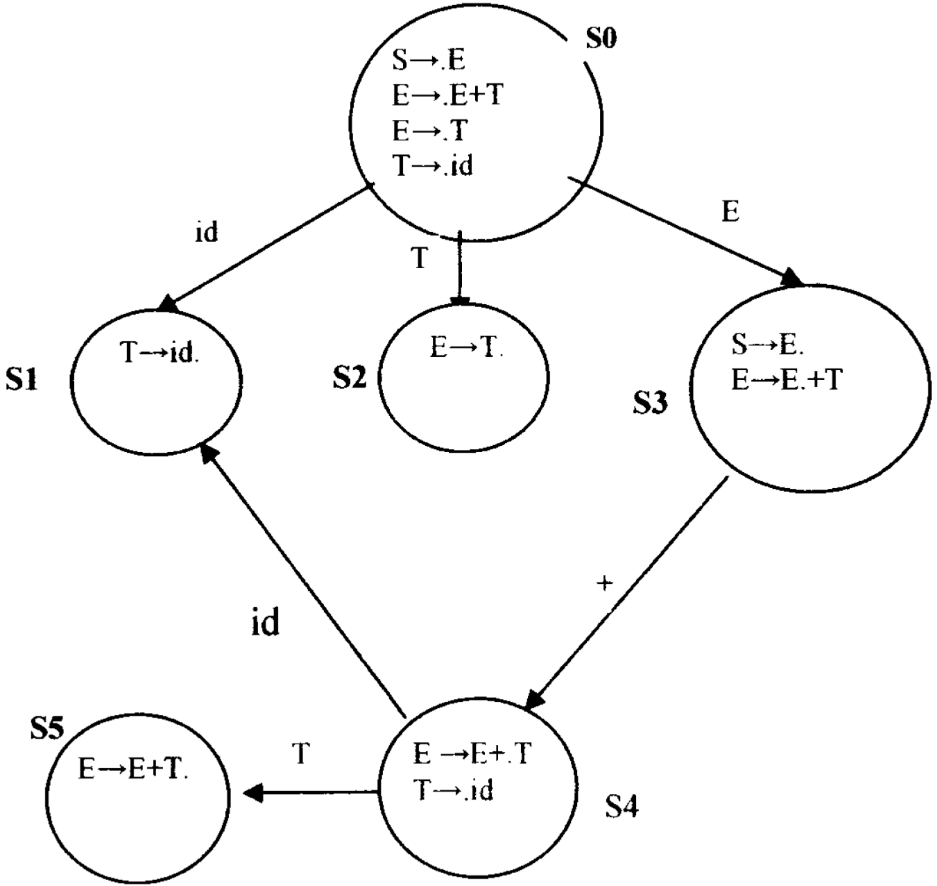
- **Example:**

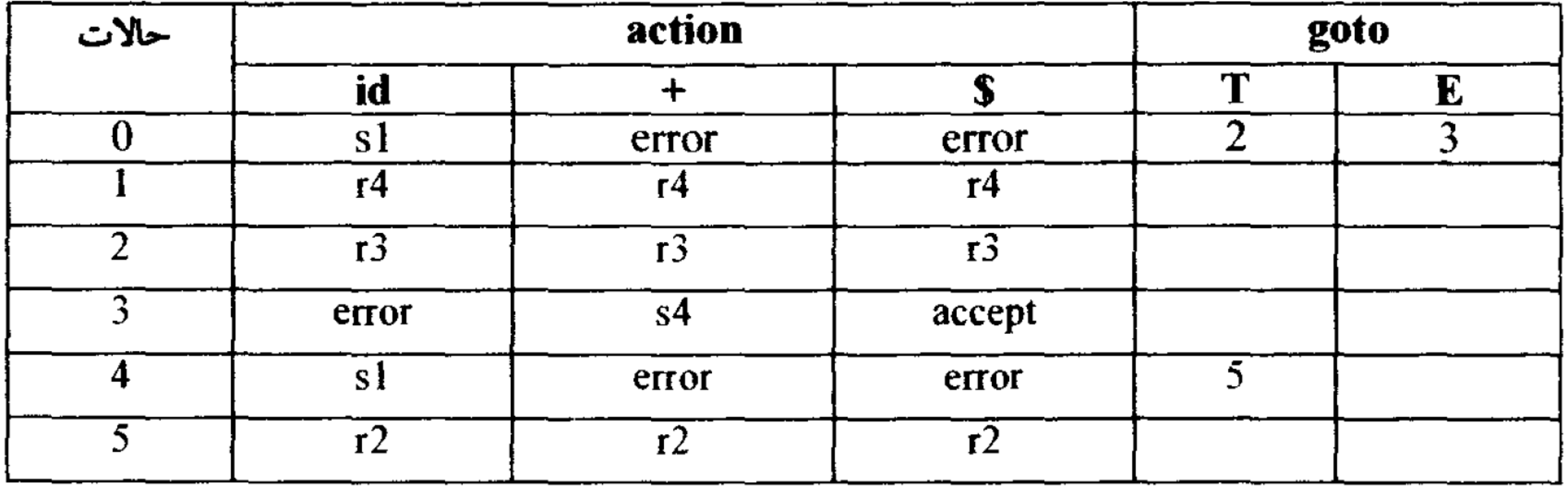
1- S→E
2- E→E+T
3- E→T
4- T→id



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)***



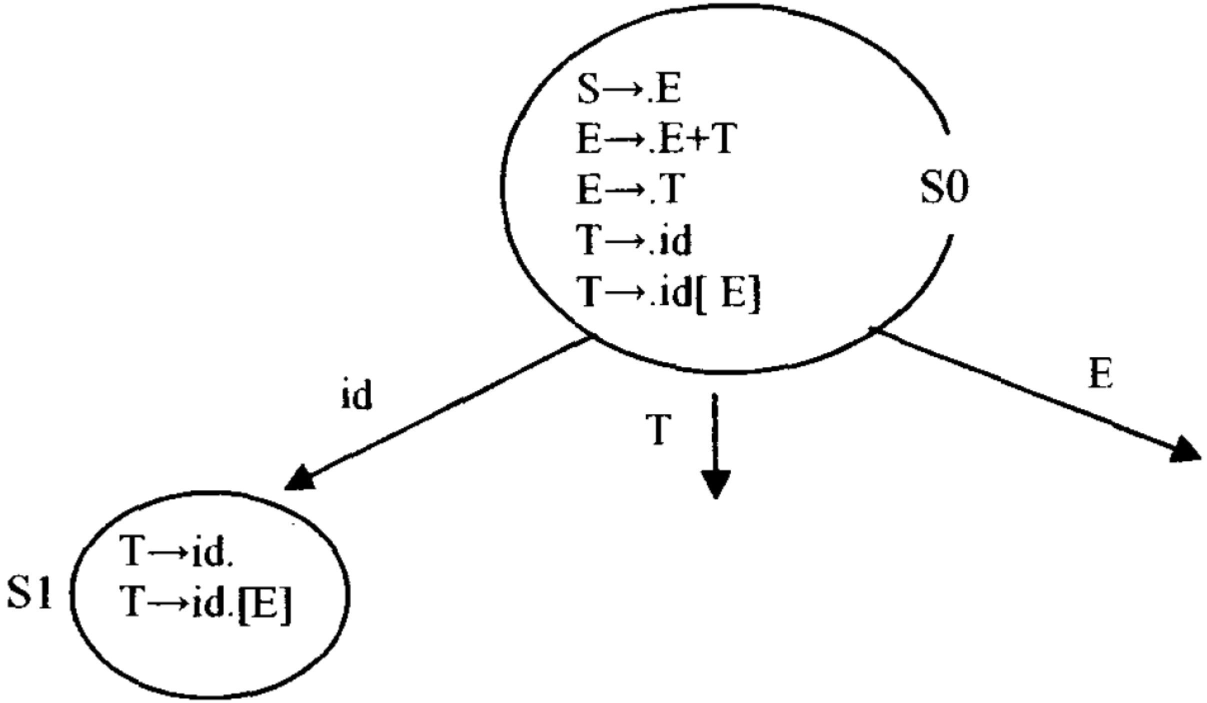


حالات	action			goto	
	id	+	$	T	E
0	s1	error	error	2	3
1	r4	r4	r4		
2	r3	r3	r3		
3	error	s4	accept		
4	s1	error	error	5	
5	r2	r2	r2		

LR(0) Grammar

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

S→E
E→E+T
E→T
T→id
T→id [E]



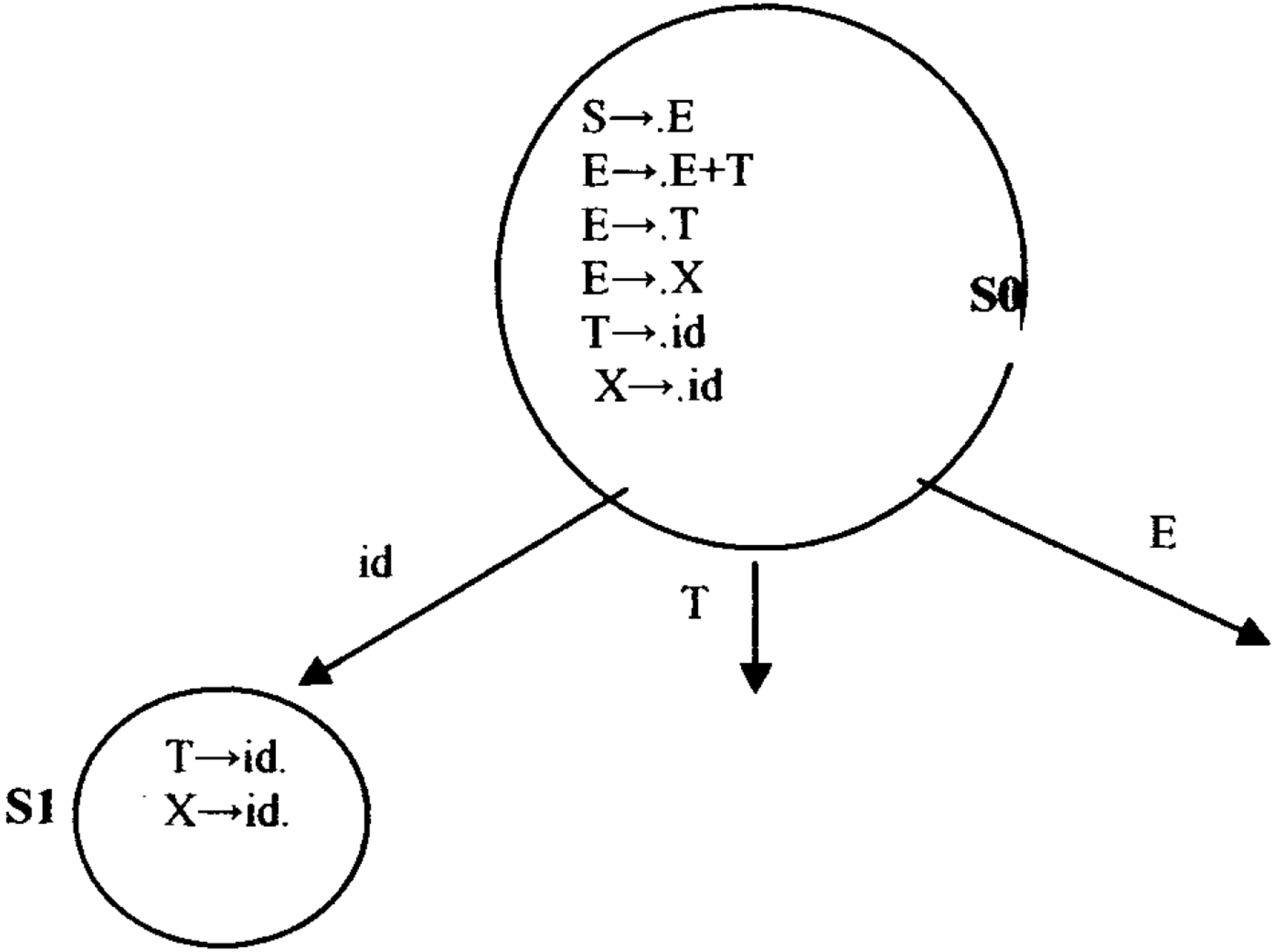
S0
S→.E
E→.E+T
E→.T
T→.id
T→.id[E]

S1
T→id.
T→id.[E]

LR(0) Grammar

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

S→E
E→E+T
E→T
E→X
T→id
X→ id



S0

S→.E
E→.E+T
E→.T
E→.X
T→.id
X→.id

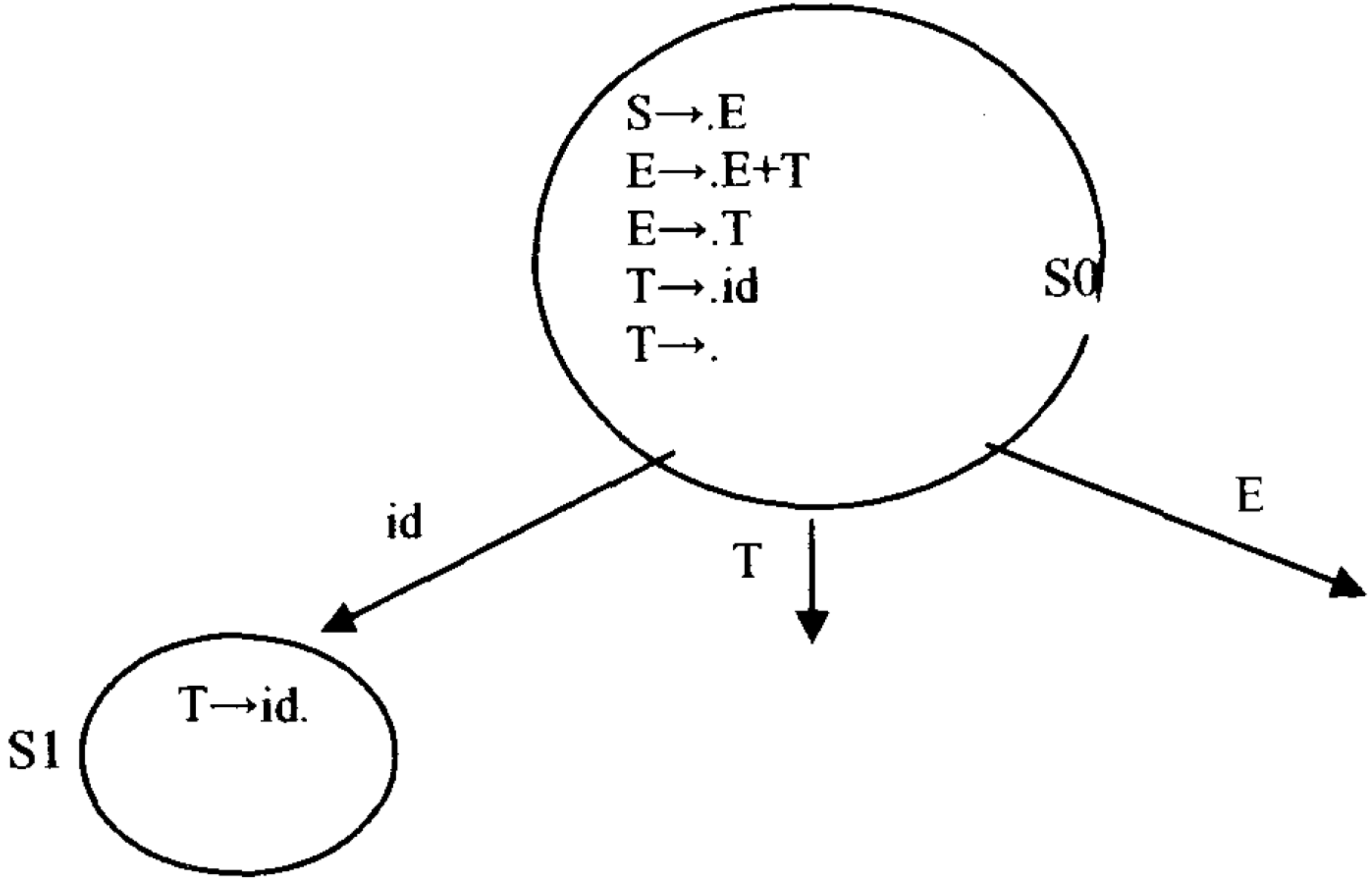
S1

T→id.
X→id.

LR(0) Grammar

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

S→E
E→E+T
E→T
T→∈
T→id



S→.E
E→.E+T
E→.T
T→.id
T→.

T→id.