

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: $S \rightarrow 01$
\$00S	11\$		Shift
\$00S1	1\$	0S1	Reduce : $S \rightarrow 0S1$
\$0S	1\$		Shift
\$0S1	\$	0S1	Reduce : $S \rightarrow 0S1$
\$S	\$		Accept

S متغیر شروع است

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

K: تعداد نمادهایی از ورودی است که نیاز داریم ببینیم

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:

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

4 تا اitem داریم که از قاعده بالا ميشه ايجاد کرد

اitem توی یک گرامر:
سمت راست یک قاعده از اون گرامر که یک
نقطه سمت راستش وجود داره که به این گفته ميشه
یک اitem

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

کلوزر: اگر نقطه قبل از نمادی باشد که اون نماد خودش نان ترمینال باشد باید قواعد مربوط به اون رو بنویسیم
گوتو:
اولین کار یک نماد جدید اضافه می کنیم --> قاعده جدید اضافه میکنیم که این قاعده جدید متغیر شروع رو میده

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**
 - *به این قاعده برسیم ینی اکسپت شده رشته*
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:

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

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

یک ایتم خاص گرفته اینجا
برای این ایتم میخواد کلوزر حساب کنه

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

Items and the LR(0) Automaton

- We divide all the sets of items of interest into two classes:
 - **Kernel items:** the initial item, $S' \rightarrow \cdot S$, 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 $S' \rightarrow \cdot S$

$E' \rightarrow \cdot E$	Kernel items	
$E \rightarrow \cdot E + T$		
$E \rightarrow \cdot T$	Non-kernel items	کرنل منشأ اون بستار است و بقیه از روی اون ساخته میشه
$T \rightarrow \cdot T * F$		
$T \rightarrow \cdot F$		
$F \rightarrow \cdot (E)$		
$F \rightarrow \cdot id$		

اون ایتm هایی که علامت دات سمت چپ اون قاعده است

Items and the LR(0) Automaton

- **The Function GOTO** ینی انتقال
 - 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

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

$$T \rightarrow \cdot T * F$$

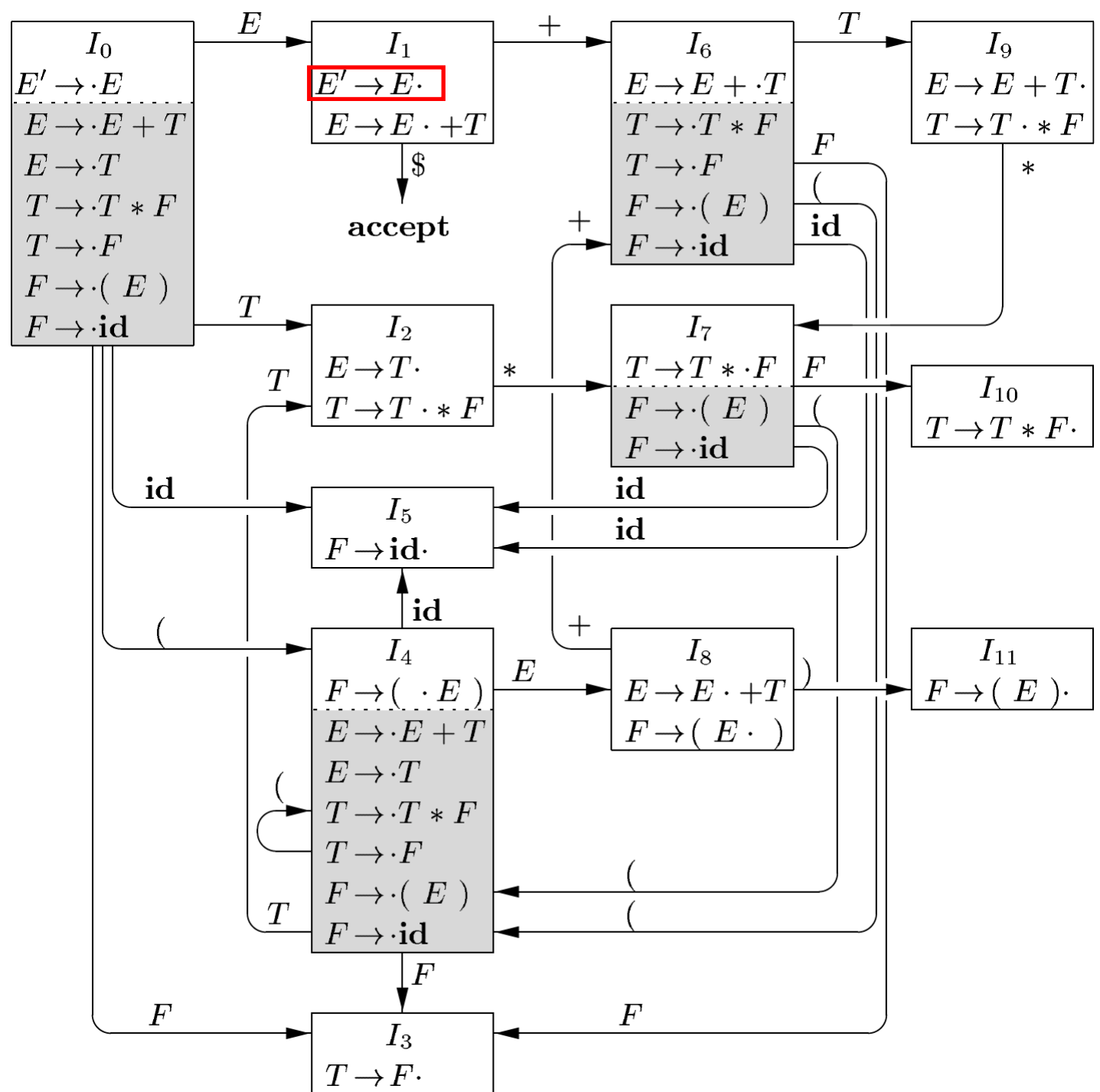
$$T \rightarrow \cdot F$$

$$F \rightarrow \cdot (E)$$

$$F \rightarrow \cdot \text{id}$$

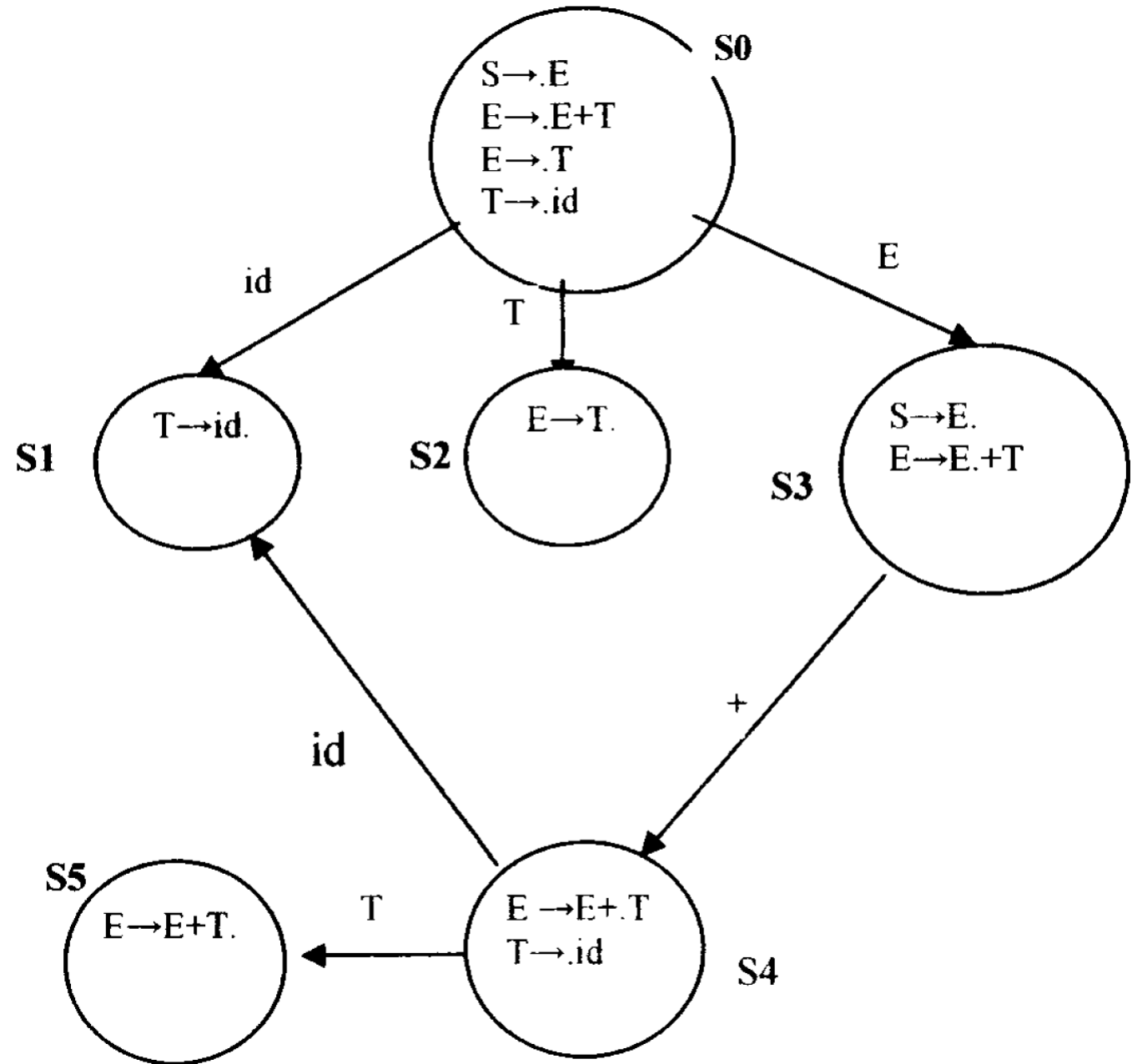
با دیدن کاراکتر جمع حرکتی انجام بشه یا نه ینی دات دقیقاً قبلش باید باشه که بتونیم حرکت انجام بدیم

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



• **Example:**

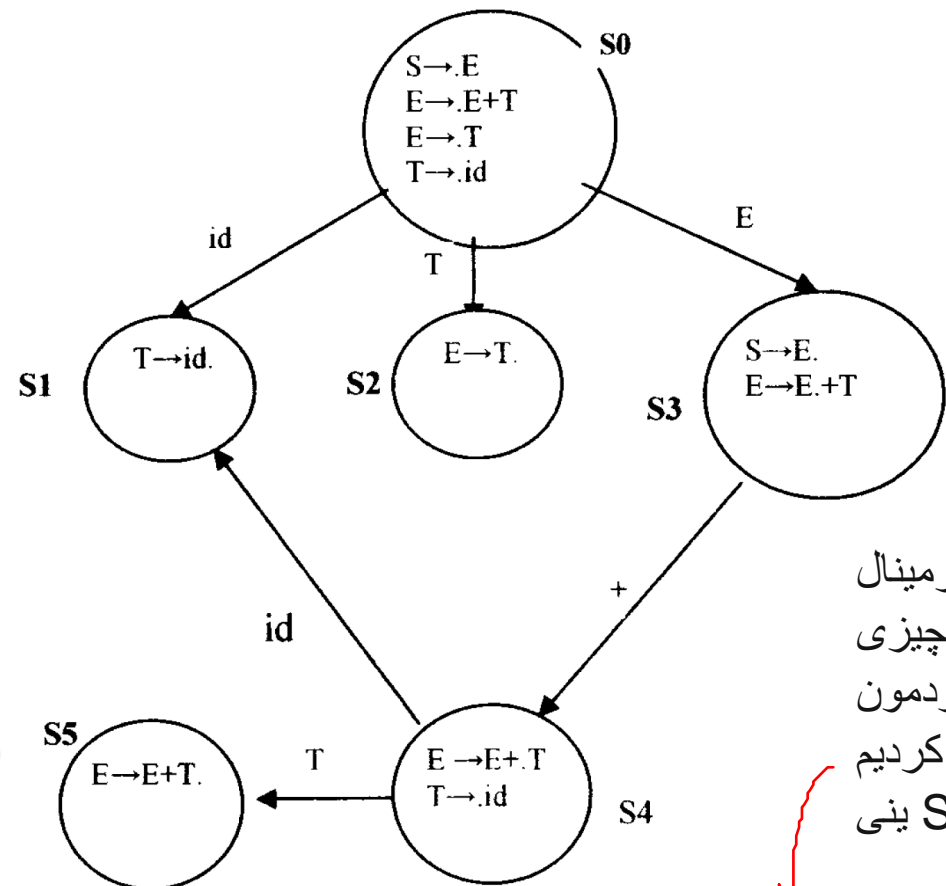
- 1- $S \rightarrow E$
- 2- $E \rightarrow E+T$
- 3- $E \rightarrow T$
- 4- $T \rightarrow id$



اگر توی یک سلول بیشتر از یک حالت بود در این حالت گرامر LR نیست

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)*



S1 ینی یه دونه شیفت داریم به استیت S 1 اول ینی شیفت خورده

نان ترمینال بدون اون چیزی که خودمون اضافه کردیم نباید باشه S ینی

بخاطر صفر ینی اصلا به هیچی از ورودی نگاه نمی کنیم و کاملاً چشم بسته توی این روش جدول رو پر میکنه بدون توجه به ورودی و این ضعیف است چون خیلی حالت تصادم پیش میاره و وقتی یک باشه ینی یه کاراکتر از ورودی در نظر میگیرن که قوی تر باشه

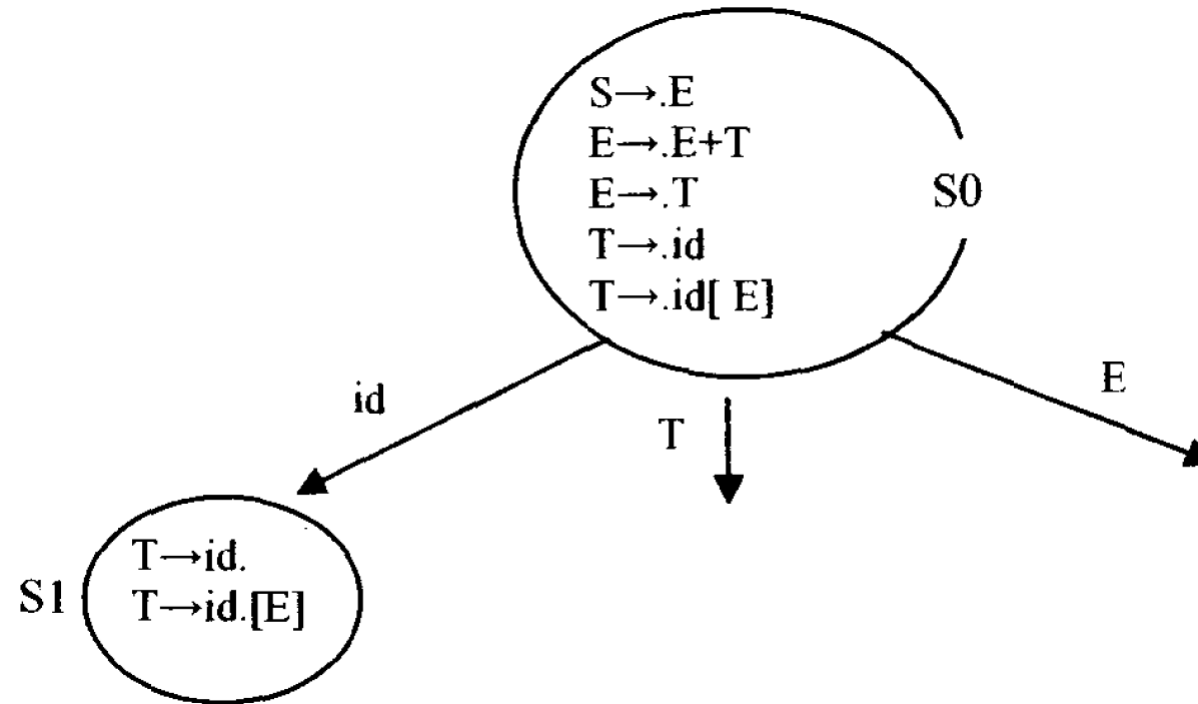
حالات	action			goto	
	id	+	\$	T	E
0	s1	error	error	2	3
1	r4	r4	r4		
2	r3	r3	r3		
3	error	s4	accept		
4	s1	error	error	5	
5	r2	r2	r2		

اول شیفت رو بررسی می کنیم و بعد کاهش رو: توی صفحه روبه رو
مثال صفحه روبه رو <--- کاهش: توی S1 هیچ یالی نداریم پس نه شیفت داریم و نه گوتو و یک قاعده داریم که رسیده به انتهایش
پس این کاهش است <-- اینجا اصلا کاری به ورودی نداره فقط می‌گه الان حالت 1 همش میشه کاهش و به ازای همه ترمینال هاش
کاهش رو می نویسه

LR(0) Grammar

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

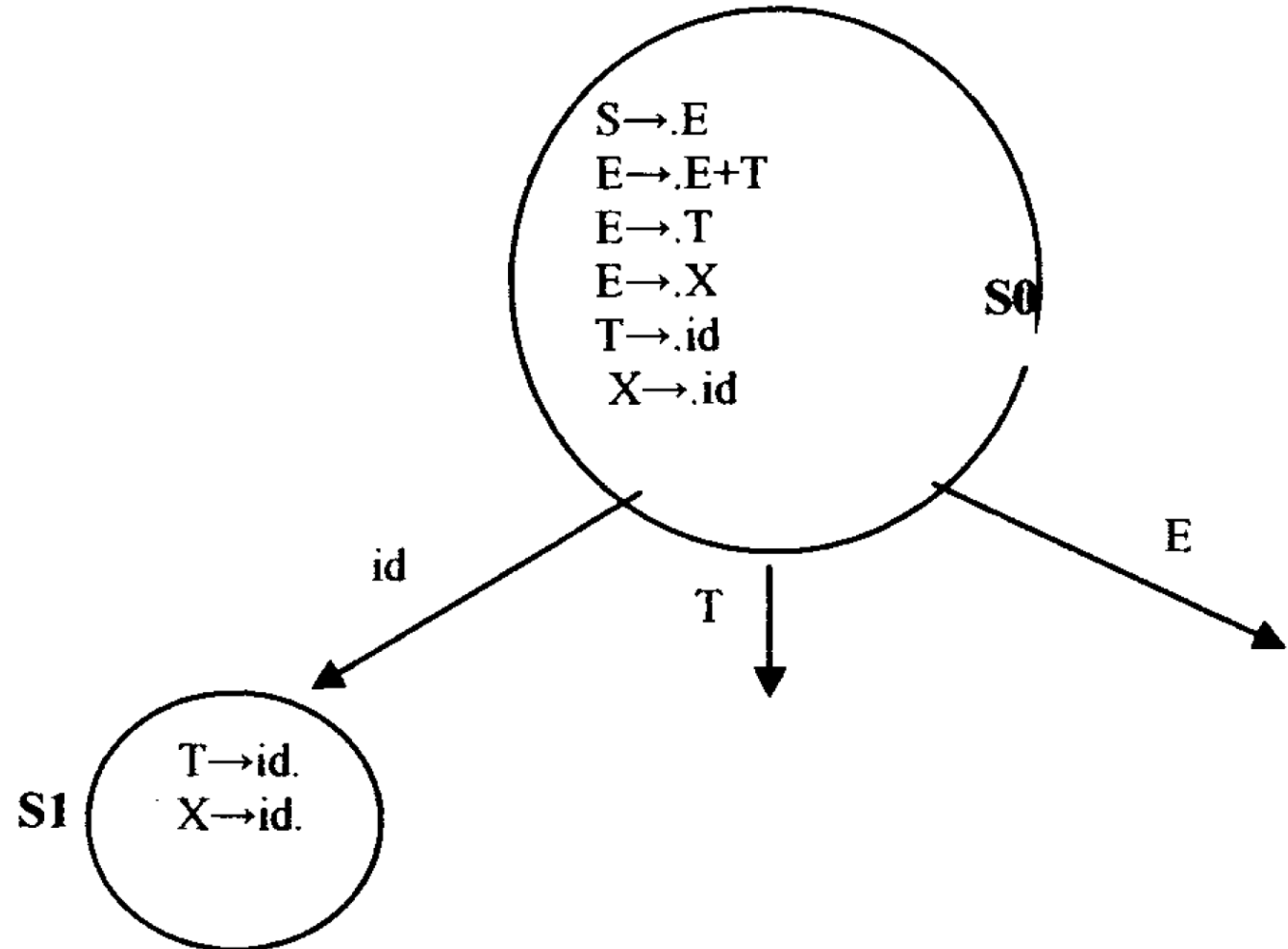
$S \rightarrow E$
 $E \rightarrow E+T$
 $E \rightarrow T$
 $T \rightarrow id$
 $T \rightarrow id [E]$



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)

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

