



# CHƯƠNG TRÌNH DỊCH

---

## **Bài 13: Phân tích LR & các bộ tự động sinh parser**



# Nội dung

---

1. Bộ phân tích kiểu gạt-thu (shift-reduce)
2. Máy phân tích cú pháp LR
3. Văn phạm họ LR
  - CLOSURE và GOTO
  - Đồ thị LR(0)
  - SLR
4. Đánh giá về phân tích LR
5. Các bộ tự động sinh parser
6. Bài tập



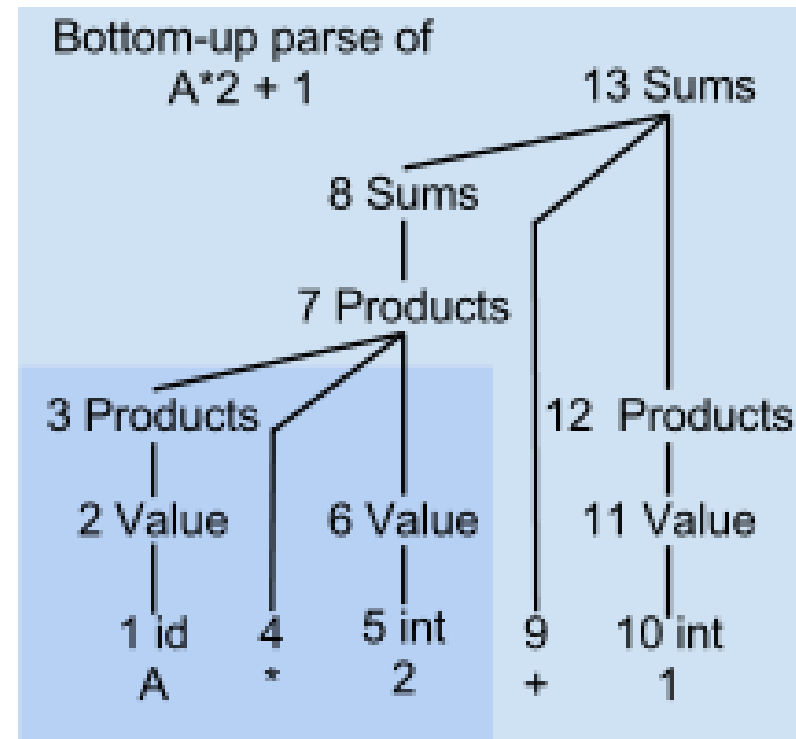
Phần 1

# Bộ phân tích kiểu gạt-thu (shift-reduce)



# Bộ phân tích kiểu gạt-thu

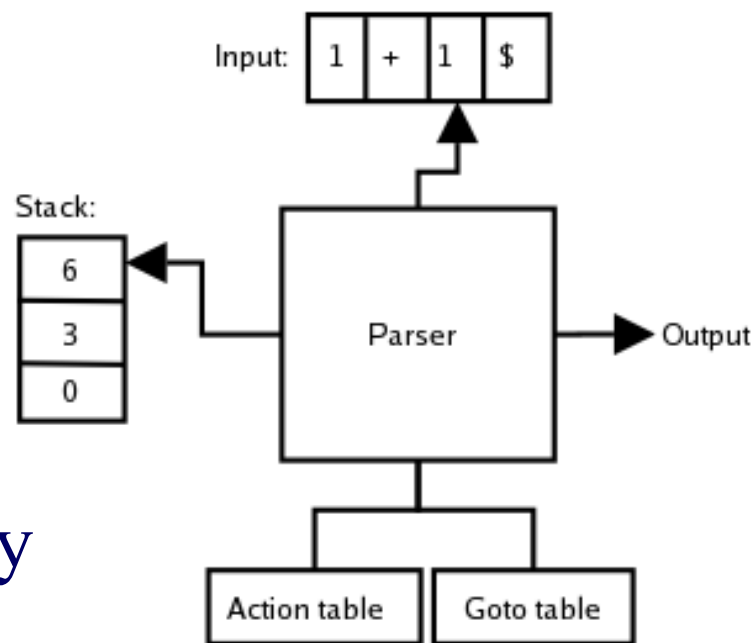
- Cách làm việc xuất phát từ việc quan sát hoạt động của phân tích bottom-up
- Bắt đầu từ nút lá phải nhất
- Thu gọn dần về nút gốc
- Chỉ 2 kiểu hoạt động chính:
  - Gạt (shift)
  - Thu (reduce)
- Shift: lấy kí hiệu tiếp theo
- Reduce: thu gọn nhánh thành một kí hiệu trung gian





# Bộ phân tích kiểu gạt-thu

- Là một dạng automat làm việc theo bảng phương án (đã được đề cập tới trong bài trước)
- Vấn đề: xây dựng bảng phương án như thế nào
  - Khi nào thì shift
  - Khi nào thì reduce
  - Còn hoạt động nào khác?
  - Có trạng thái bị tranh chấp?
- Hoạt động của stack ra sao?
- Ý nghĩa các trạng thái của máy



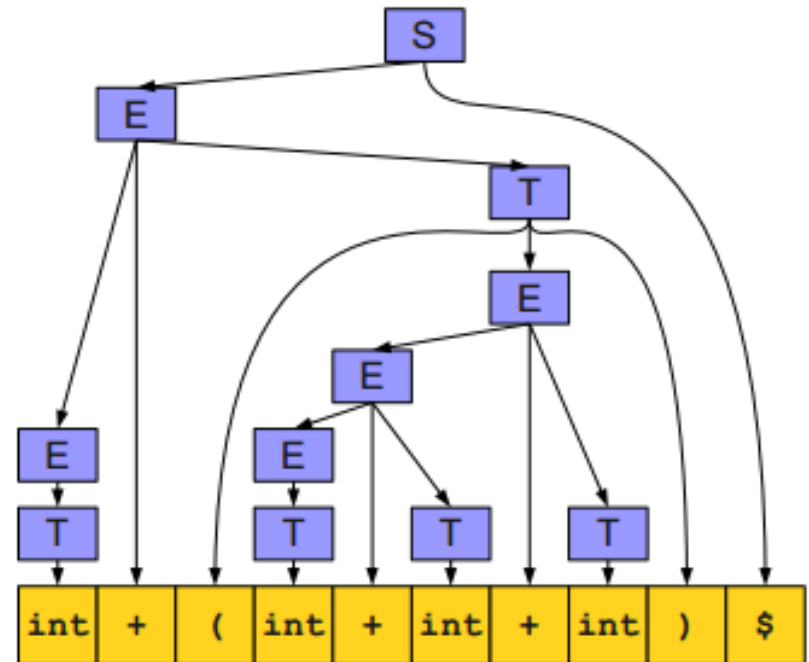


# Ví dụ về bộ phân tích gọt-thu

Grammar:

- $S \rightarrow E\$$
- $E \rightarrow T$
- $E \rightarrow E + T$
- $T \rightarrow \text{int}$
- $T \rightarrow ( E )$

*int + (int + int + int)*





# Ví dụ về bộ phân tích gọt-thu

Grammar:

$S \rightarrow E$

$E \rightarrow T$

$E \rightarrow E + T$

$T \rightarrow \mathbf{int}$

$T \rightarrow ( E )$

*int* + (*int* + *int* + *int*)\$

*T* + (*int* + *int* + *int*)\$

*E* + (*int* + *int* + *int*)\$

*E* + (*T* + *int* + *int*)\$

*E* + (*E* + *int* + *int*)\$

*E* + (*E* + *T* + *int*)\$

*E* + (*E* + *int*)\$

*E* + (*E* + *T*)\$

*E* + (*E*)\$

*E* + *T*\$

*E*\$

*S*



# Ví dụ về bộ phân tích gạt-thu

Grammar:

$$E \rightarrow E + T | T$$

$$T \rightarrow T * F | F$$

$$F \rightarrow ( E ) | id$$

Bottom-up parsing of  
 $id + id * id$

Left substring	Right substring	Action
\$	$id + id * id$ \$	Shift
$\$id$	$+id * id$ \$	Reduce by $F \rightarrow id$
$\$F$	$+id * id$ \$	Reduce by $T \rightarrow F$
$\$T$	$+id * id$ \$	Reduce by $E \rightarrow T$
$\$E$	$+id * id$ \$	Shift
$\$E+$	$id * id$ \$	Shift
$\$E + id$	$*id$ \$	Reduce by $F \rightarrow id$
$\$E + F$	$*id$ \$	Reduce by $T \rightarrow F$
$\$E + T$	$*id$ \$	Shift
$\$E + T*$	$id$ \$	Shift
$\$E + T * id$	\$	Reduce by $F \rightarrow id$
$\$E + T * F$	\$	Reduce by $T \rightarrow T * F$
$\$E + T$	\$	Reduce by $E \rightarrow E + T$
$\$E$	\$	Accept



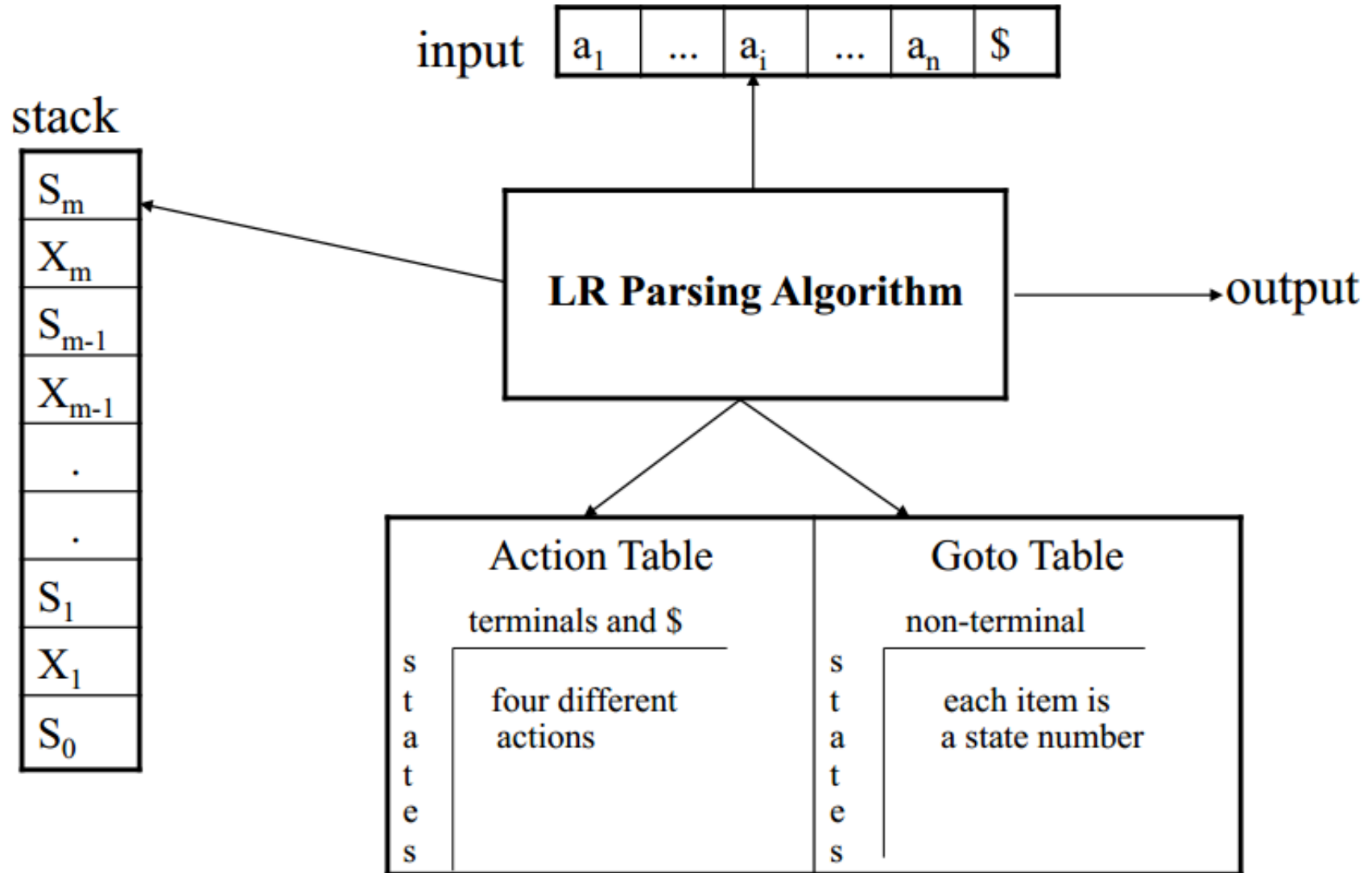


Phần 2

# Máy phân tích cú pháp LR



# Cấu trúc của máy phân tích LR





# Cấu trúc của máy phân tích LR

---

- Máy phân tích LR là một cặp (STACK, INPUT)
  - Trạng thái ban đầu ( $s_0, a_1a_2\dots a_n\$$ )
  - Trạng thái trung gian ( $s_0X_1s_1\dots X_ms_m, a_ia_{i+1}\dots a_n\$$ )
  - Trạng thái kết thúc thành công ( $s_0Ss_1, \$$ )
- Bảng phương án gồm 2 phần
  - Bảng action: ACTION[s, a] với s là một trạng thái và a là một terminal, giá trị trong bảng chỉ có thể là 1 trong 4 hành động gạt (shift), thu (reduce), nhận (accept), lỗi (error)
  - Bảng goto: GOTO[s, A] với s là một trạng thái và A là một non-terminal, chỉ ra cách dịch chuyển trạng thái



# Bảng ACTION

1. Shift s: đẩy kí hiệu input và trạng thái s vào stack  
 $(s_0 X_1 s_1 \dots X_m s_m, a_i a_{i+1} \dots a_n \$) \rightarrow (s_0 X_1 s_1 \dots X_m s_m a_i s, a_{i+1} \dots a_n \$)$
2. Reduce k: thu gọn bởi luật thứ k ( $A \rightarrow \beta$ ),  $r = |\beta|$ 
  - Lấy 2r kí hiệu ra khỏi stack:  $(s_0 X_1 s_1 \dots X_m s_m, a_i a_{i+1} \dots a_n \$) \rightarrow (s_0 X_1 s_1 \dots X_{m-r} s_{m-r}, a_{i+1} \dots a_n \$)$
  - Đẩy vào stack A và  $s = \text{GOTO}[s_{m-r}, A]$ :  
 $(s_0 X_1 s_1 \dots X_{m-r} s_{m-r}, a_i a_{i+1} \dots a_n \$) \rightarrow (s_0 X_1 s_1 \dots X_m s_m A s, a_{i+1} \dots a_n \$)$
  - Ghi nhận phát sinh luật  $A \rightarrow \beta$
3. Accept: phân tích thành công
4. Error: phân tích phát hiện lỗi



# Ví dụ hoạt động của máy LR

1.  $E \rightarrow E + T$
2.  $E \rightarrow T$
3.  $T \rightarrow T * F$
4.  $T \rightarrow F$
5.  $F \rightarrow (E)$
6.  $F \rightarrow \mathbf{id}$

Action Table							Goto Table		
state	id	+	*	(	)	\$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			



# Ví dụ hoạt động của máy LR

<u>stack</u>	<u>input</u>	<u>action</u>	<u>output</u>
0	id*id+id\$	shift 5	
0id5	*id+id\$	reduce by $F \rightarrow id$	$F \rightarrow id$
0F3	*id+id\$	reduce by $T \rightarrow F$	$T \rightarrow F$
0T2	*id+id\$	shift 7	
0T2*7	id+id\$	shift 5	
0T2*7id5	+id\$	reduce by $F \rightarrow id$	$F \rightarrow id$
0T2*7F10	+id\$	reduce by $T \rightarrow T * F$	$T \rightarrow T * F$
0T2	+id\$	reduce by $E \rightarrow T$	$E \rightarrow T$
0E1	+id\$	shift 6	
0E1+6	id\$	shift 5	
0E1+6id5	\$	reduce by $F \rightarrow id$	$F \rightarrow id$
0E1+6F3	\$	reduce by $T \rightarrow F$	$T \rightarrow F$
0E1+6T9	\$	reduce by $E \rightarrow E + T$	$E \rightarrow E + T$
0E1	\$	accept	



Phần 3

# Văn phạm họ LR



# Văn phạm họ LR

---

- Việc chính là làm thế nào để xây dựng bảng phương án? Có nhiều thuật toán làm việc này
- LR(0): thuật toán cơ bản, mọi thuật toán LR đều dựa trên nó
- SLR (Simple LR): cải tiến một chút từ LR(0), mạnh hơn, dễ cài đặt
- LR(1): còn gọi là LR chính tắc ~ Canonical LR, sử dụng cho nhiều loại văn phạm, kích cỡ bảng rất lớn
- LALR(1): cân bằng giữa SLR và LR, đủ dùng cho hầu hết các văn phạm nhân tạo





# Khái niệm cơ sở

---

- Để dễ dàng cho việc thực thi automat, ta bổ sung thêm luật  $S' \rightarrow S$  vào tập luật
- Khái niệm LR(0) item: một luật đang được phân tích dở, sử dụng dấu chấm (.) để ngăn giữa phần trước và phần sau (tương tự như thuật toán Earley)
- Luật  $S \rightarrow ABC$  sẽ có 4 item:
  1.  $S \rightarrow .ABC$
  2.  $S \rightarrow A.BC$
  3.  $S \rightarrow AB.C$
  4.  $S \rightarrow ABC.$



# Closure(I) – bao đóng của I

---

CLOSURE( $I$ )

**repeat**

**for** any item  $A \rightarrow \alpha.X\beta$  in  $I$

**for** any production  $X \rightarrow \gamma$

$I = I \cup \{X \rightarrow \cdot\gamma\}$

**until**  $I$  does not change

**return**  $I$

Example:

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

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



# GOTO(I, X) – hàm chuyển

GOTO( $I, X$ )

Set  $J$  to the empty set

for any item  $A \rightarrow \alpha.X\beta$  in  $I$

$$J = J \cup \{A \rightarrow \alpha X.\beta\}$$

return CLOSURE( $J$ )

Example:

$E' \rightarrow E$	$l_0 = \{E' \rightarrow .E,$	$\text{GOTO}(l_0, E) = \{E' \rightarrow E., E \rightarrow E. + T\}$
$E \rightarrow E + T$	$E \rightarrow .E + T$	$\text{GOTO}(l_0, T) = \{E \rightarrow T., T \rightarrow T.*F\}$
$E \rightarrow T$	$E \rightarrow .T$	$\text{GOTO}(l_0, F) = \{T \rightarrow F.\}$
$T \rightarrow T * F$	$T \rightarrow .T * F$	$\text{GOTO}(l_0, '()) = \text{CLOSURE}(\{F \rightarrow (.E)\})$
$T \rightarrow F$	$T \rightarrow .F$	$= \{F \rightarrow (.E)\} \cup (l_0 \setminus \{E' \rightarrow E\})$
$F \rightarrow (E)$	$F \rightarrow .(E)$	$\text{GOTO}(l_0, \text{id}) = \{F \rightarrow \text{id}.\}$
$F \rightarrow \text{id}$	$F \rightarrow .\text{id} \}$	



# Xây dựng đồ thị LR(0)

Initialize  $T$  to  $\{\mathbf{Closure}(\{S' \rightarrow .S\})\}$

Initialize  $E$  to empty.

**repeat**

**for** each state  $I$  in  $T$

**for** each item  $A \rightarrow \alpha.X\beta$  in  $I$

**let**  $J$  be **Goto**( $I, X$ )

$T \leftarrow T \cup \{J\}$

$E \leftarrow E \cup \{I \xrightarrow{X} J\}$

**until**  $E$  and  $T$  did not change in this iteration

$R \leftarrow \{\}$

**for** each state  $I$  in  $T$

**for** each item  $A \rightarrow \alpha.$  in  $I$

$R \leftarrow R \cup \{(I, A \rightarrow \alpha)\}$

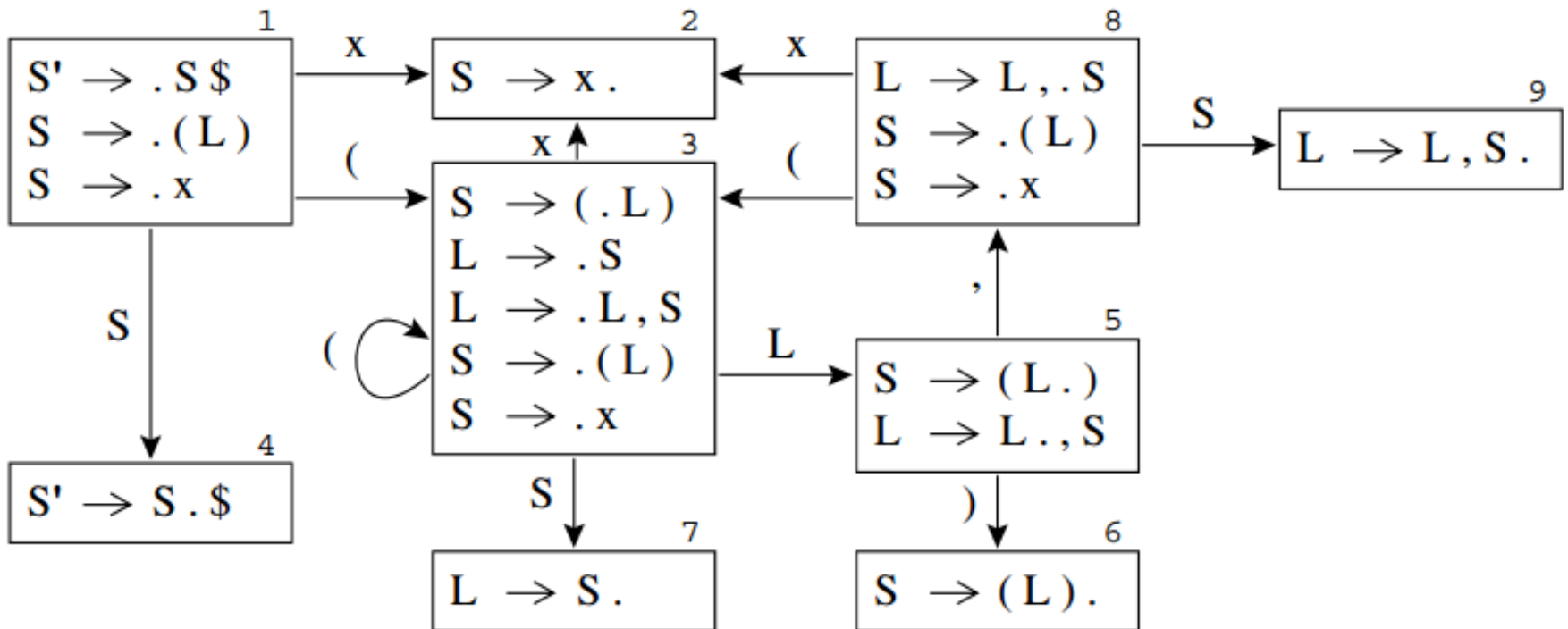
- Có cạnh  $I$  đến  $J$  là  $X$ 
  - $X$  là terminal:  $(I, X) = \text{shift } J$
  - $X$  là non-terminal:  $(I, X) = \text{goto } J$
  - Nếu  $X = \$$ : accept
- Nếu state chứa  $A \rightarrow \beta.$  thì điền reduce vào mọi ô trên dòng



# Ví dụ

$S' \rightarrow S \$$        $S \rightarrow (L)$        $S \rightarrow x$

$L \rightarrow S$        $L \rightarrow L, S$





# Ví dụ

---

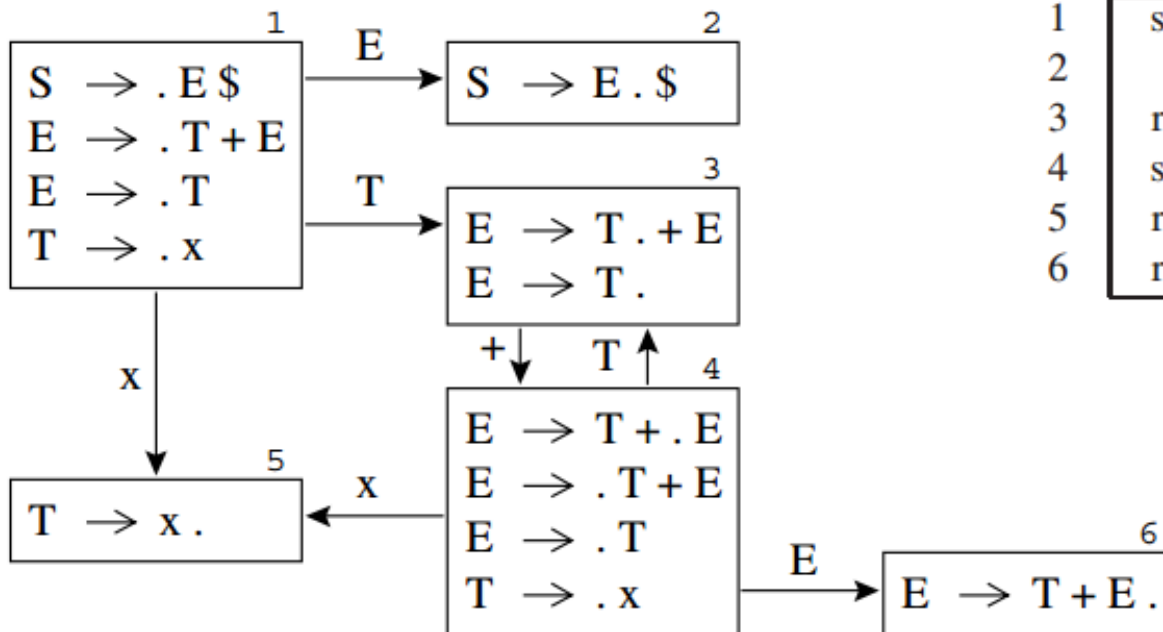
	(	)	x	,	\$	<i>S</i>	<i>L</i>
1	s3		s2			g4	
2	r2	r2	r2	r2	r2		
3	s3		s2			g7	g5
4					a		
5		s6		s8			
6	r1	r1	r1	r1	r1		
7	r3	r3	r3	r3	r3		
8	s3		s2			g9	
9	r4	r4	r4	r4	r4		



# SLR

$S' \rightarrow E \$$        $E \rightarrow T + E$

$E \rightarrow T$            $T \rightarrow x$



	x	+	\$	E	T
1	s5			g2	g3
2			a		
3	r2	s4,r2	r2		
4	s5			g6	g3
5	r3	r3	r3		
6	r1	r1	r1		



# SLR

---

- SLR sửa đổi lại cách tính reduce, chỉ sử dụng reduce cho những tình huống  $X$  thuộc  $\text{FOLLOW}(A)$
- Chú ý: có nhiều loại xung đột, phương pháp SLR chỉ sửa được một phần rất nhỏ
  - Xung đột giữa shift và reduce
  - Xung đột giữa reduce và reduce

$R \leftarrow \{\}$

**for** each state  $I$  in  $T$

**for** each item  $A \rightarrow \alpha.$  in  $I$

**for** each token  $X$  in  $\text{FOLLOW}(A)$

$R \leftarrow R \cup \{(I, X, A \rightarrow \alpha)\}$





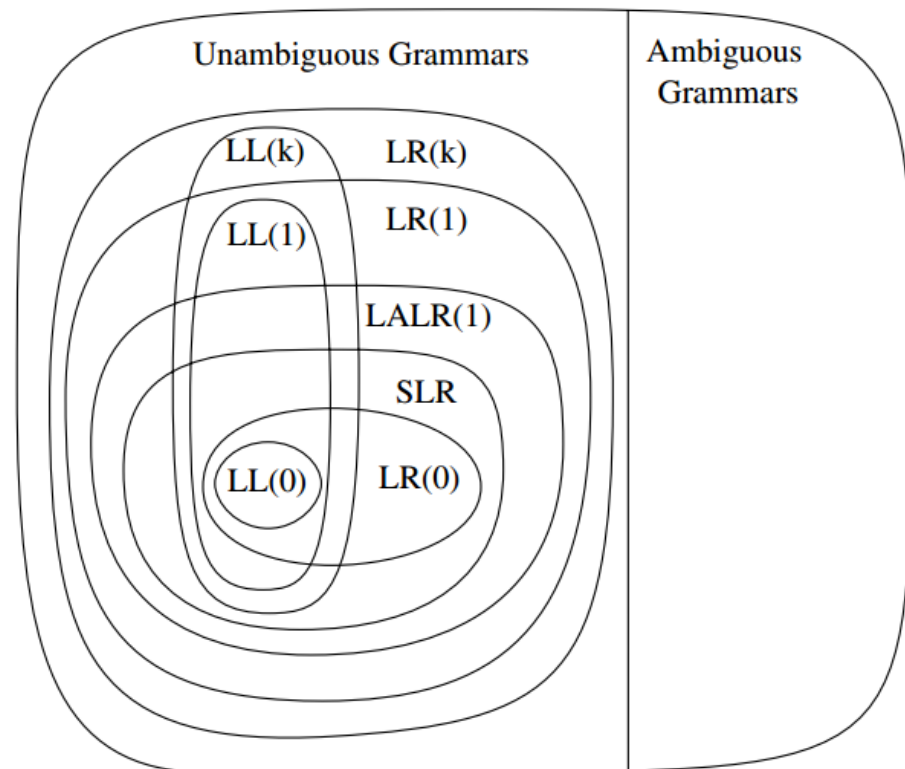
Phần 4

# Đánh giá về phân tích LR



# Đánh giá về phân tích LR

- Phân tích LR không đủ mạnh cho văn phạm CFG
- Nhưng đủ mạnh cho hầu hết ngôn ngữ nhân tạo
  - LR(0): là hạt nhân
  - SLR: đơn giản, yếu
  - LALR(1): tạm đủ dùng
  - LR(1): bảng quá to
  - LR(k): quá phức tạp
- Nhanh ~ tuyến tính
- Rất nhiều biến thể
- GLR ~ Earley





Phần 5

# Các bộ tự động sinh parser



# Các bộ tự động sinh parser

---

- Với cách tiếp cận xây dựng automat tất định: cho trước văn phạm  $G$ , ta có thể tạo một bảng phân tích riêng của  $G$ , bảng phân tích này chỉ cần tạo một lần và cố định đối với văn phạm  $G$
- Các bộ parser generator tự động hóa việc xây dựng các bộ phân tích văn phạm:
  - Người dùng định nghĩa văn phạm  $G$
  - Thiết lập các xử lý cần thực hiện khi hoàn thành câu
  - Phần mềm phân tích  $G$ , tự sinh bảng phương án
  - Phần mềm tự sinh mã bộ phân tích, chèn những đoạn xử lý vào các vị trí thích hợp



# Các bộ tự động sinh parser

---

- Hầu hết các parser generator sinh bảng LALR(1)
  - Bảng này đủ tốt để xử lý hầu hết các ngôn ngữ nhân tạo
  - Bảng kích thước không quá lớn (với ngôn ngữ C, bảng LR(1) có khoảng 10000 trạng thái, bảng LALR chỉ có khoảng 350 trạng thái)
- Parser generator đầu tiên là META II (1960)
- Nổi tiếng nhất: YACC (1975, mã C)
- Sinh mã Java: SableCC
- Sinh mã C#, giao diện trực quan: GOLD Parser  
*(yêu cầu tìm hiểu phần mềm này như là bài tập)*



Phần 6

# Bài tập



# Bài tập

---

1. Cho văn phạm  $G$ :

$$S \rightarrow AS \mid b \qquad A \rightarrow SA \mid a$$

- Xây dựng bộ các tập item  $LR(0)$  cho văn phạm này
- Xây dựng bảng phân tích cú pháp bằng thuật toán SLR

2. Cho văn phạm  $G$ :

$$E \rightarrow E + T \mid T$$

$$T \rightarrow TF \mid F$$

$$F \rightarrow F^* \mid a \mid b$$

- Xây dựng bộ các tập item  $LR(0)$  cho văn phạm này
- Xây dựng bảng phân tích cú pháp bằng thuật toán SLR