

## 컴파일러: 6장

국민대학교 컴퓨터공학부  
강 승 식

1

## 제6장 Bottom-up 구문 분석

- LR 파싱
  - Left-to-right scanning & Rightmost derivation
  - Deterministic parsing
  - *우단 역유도* 방식에 따라 *결정적인* 방법으로 파싱
- 우단유도에서 비결정성 문제
  - Shift-reduce conflict
  - Reduce-reduce conflict
- 백트래킹이 없는 결정적 파싱이 가능하려면...
  - 각 단계에서 shift할지, reduce할지 결정되어야 함
  - Reduce할 경우 어떤 생성규칙으로 reduce할지 결정되어 있어야 함

2

## LR(1) 조건, LR(1) 문법

- LR(1) 조건
  - 우단 역유도의 각 단계에서 현재 위치에서 reduce되어야 할 입력(토큰) 1개를 보고 shift할지, reduce할지를 알 수 있고, reduce할 경우 어떤 생성규칙을 적용할지 알 수 있는 결정적 파싱이 가능한 문법
- LR(1) 문법
  - LR(1) 조건을 만족하는 문법

3

## LR(k) 문법

- LR(k) 문법은 현재 위치에서 생성해야 할  $d$  입력(토큰)  $k$ 개를 참조하여 결정적 파싱이 가능한 문법

4

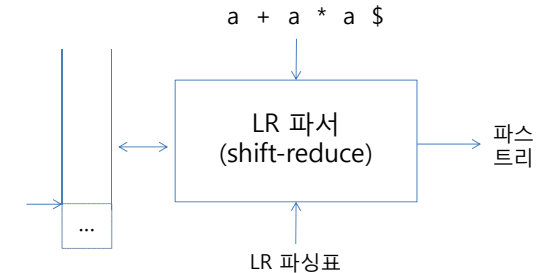
## LR 조건을 만족하지 않는 문법

- 파싱표 작성할 때 <state, 터미널> 항에 shift-reduce 혹은 reduce-reduce conflict가 발생하는 문법은 LR 조건을 만족하지 않음
- LR 조건을 만족하지 않는 문법은 그 의미에 따라 강제로 하나의 생성규칙을 선택한다면 결정적인 파서를 구현할 수 있다

5

## LR 파서의 구조

- Shift: 입력 토큰 1개를 스택으로 이동
- Reduce: 스택의 top 부분에서 어떤 생성규칙의 RHS와 일치되는 부분을 LHS로 대치



6

## LR 파서의 작동 방식

- 스택 내용을 시작 상태( $S_0$ )로 초기화
- 스택의 top에 있는 '현재 상태 번호'와 입력 버퍼의 터미널 토큰에 대해 LR 파싱표에 지정되어 있는 연산을 수행
- shift 연산의 경우
  - 입력 버퍼의 터미널 토큰을 스택으로 이동하고,
  - 파싱표에 지정된 다음 상태 번호를 스택에 삽입
- reduce 연산의 경우
  - 파싱표에 지정된 생성규칙 번호에 의해 reduce를 수행
  - 생성규칙의 LHS와 함께 '다음 상태 번호'를 삽입
    - 스택의 <상태번호, 삽입된 논터미널 기호>에 대해 GOTO 테이블에 지정된 상태 번호를 스택에 삽입

7

## LR 파서

- LR 파서의 스택 내용
  - 초기 상태 번호로 시작
    - 스택의 top 기호는 항상 상태 번호
  - Shift 연산에 의해
    - <입력토큰, next state> 쌍을 스택에 push
  - Reduce 연산에 의해
    - RHS에 해당하는 <문법기호, state> 쌍들을 pop
    - LHS에 해당하는 <문법기호, next state> push

8

## LR 파싱표

기호	ACTION 테이블	GOTO 테이블
상태	$t_1 t_2 t_3 \dots t_n \$$	$N_1 N_2 N_3 \dots N_m$
0	<shift, 다음 상태 번호>	다음 상태 번호
1	<reduce, 생성규칙 번호>	
2	<accept>	
...	<error>	

9

## LR 파싱 예제

- $E \rightarrow E + F$
- $E \rightarrow F$
- $F \rightarrow a$

a+a에 대한 파싱 과정

심별 상태	ACTION 테이블			GOTO 테이블	
	a	+	\$	E	F
0	s3			1	2
1		s4	acc		
2		r2	r2		
3		r3	r3		
4	s3				5
5		r1	r1		

스택	입력 버퍼	동작
0	a + a \$	s3
0 a 3	+ a \$	r3 GOTO 2
0 F 2	+ a \$	r2 GOTO 1
0 E 1	+ a \$	s4
0 E 1 + 4	a \$	s3
0 E 1 + 4 a 3	\$	r3 GOTO 5
0 E 1 + 4 F 5	\$	r1 GOTO 1
0 E 1	\$	accept

<과제> 위 LR 파싱표와 LR 파서를 구현하시오.

10

## LR 파싱표 작성

- LR(0) item 집합과 오토마타 구성
  - 주어진 문법에 대한 오토마타를 구성
    - 오토마타의 각 상태는 LR(0) item 집합
- LR(0) item
  - 각 생성규칙의 RHS 부분에 dot 표시  
생성규칙을 대괄호로 둘러쌘
  - Mark 심별
    - dot 표시가 있는 터미널 또는 논터미널 기호
    - 예)  $[A \rightarrow \cdot BCD]$  에서 mark 심별은 B

11

## LR(0) item

- LR(0) item 유형
  - Closure item, kernel item, reduce item
- 생성규칙  $A \rightarrow BCD$ 에 대한 LR(0) item
  - $[A \rightarrow \cdot BCD]$  : closure item
  - $[A \rightarrow B \cdot CD]$  : kernel item
  - $[A \rightarrow BC \cdot D]$  : kernel item
  - $[A \rightarrow BCD \cdot]$  : reduce item

12

## LR(0) item의 의미

- LR(0) item은
  - 생성규칙의 RHS 부분이 스택에서 reduce될 준비가 되어 있는지를 나타낸다.
  - 즉, 생성규칙의 RHS가 스택의 top 부분에 현재 어디까지 완성되어 있는지를 보여준다.
- 예)
  - $[A \rightarrow BC \cdot D]$  : 현재 BC가 스택의 top 부분에 있고, D가 스택에 삽입되면 reduce될 수 있음
  - $[A \rightarrow BCD \cdot]$  : reduce할 준비가 되어 있음

13

- Closure item
  - dot 위치가 RHS의 맨 앞
- Kernel item
  - dot 위치가 RHS의 터미널/논터미널 중간에 있는 LR(0) item
- Reduce item
  - dot 위치가 RHS의 맨 끝, reduce 준비 완료

14

## Closure item

- Mark 기호가 non-terminal이면, 그 생성규칙으로 reduce가 되어 있어야 한다
- 예)  $[A \rightarrow \cdot BCD]$  의 경우
  - 논터미널 B가 스택에 삽입되려면 B-생성규칙이 먼저 적용되어야 함.
  - 따라서 B-생성규칙으로 reduce될 수 있도록  $[A \rightarrow \cdot BCD]$ 에 대한 CLOSURE 집합을 구성

15

## CLOSURE 집합

- $CLOSURE([A \rightarrow \cdot BCD]) =$ 
  - $\{ [A \rightarrow \cdot BCD] \} \cup \{ \text{mark 심벌이 논터미널인 생성규칙들에 대한 closure item} \}$
- 예)  $B \rightarrow CD, B \rightarrow EF$  에 대해
  - $CLOSURE([A \rightarrow \cdot BCD]) =$ 
    - $\{ [A \rightarrow \cdot BCD], [B \rightarrow \cdot CD], [B \rightarrow \cdot EF] \} \cup$
    - $\{ \text{C-생성규칙과 E-생성규칙에 대한 closure item} \}$
- 시작기호에 대한 CLOSURE 집합이 오토마타의 초기상태가 된다

16

## LR 파싱표를 위한 오토마타 구성

- 1) 문법에 새로운 시작기호 추가:  $ACC \rightarrow S$
- 2) 초기 상태  $S_0$   
새로운 시작기호의 closure item에 대한 CLOSURE 집합  
 $CLOSURE([ACC \rightarrow \cdot S]) = \{ [ACC \rightarrow \cdot S], [S \rightarrow \cdot \alpha], \dots \}$
- 3) 다음 상태 구성  
- 각 상태의 CLOSURE 집합에서 각 mark 심벌들에 대해 dot 위치를 mark 심벌의 다음 위치로 이동한 LR(0) item에 대해 CLOSURE 집합 구성
- 4) 더 이상 새로운 상태가 만들어지지 않을 때까지 반복

17

## 오토마타 구성

- 현재 상태  $S_i$ 에서 mark 기호  $X$ 에 대해 다음 상태를  $S_j$ 로 구성  
-  $GOTO(S_i, X) = S_j$   
- 다음 상태의 개수는? mark 기호의 개수
- 초기 상태에서부터 시작하여 각 상태에서  $X$ -transition에 의해 다음 상태 구성

18

## 오토마타 구성 예제

### (1) 새로운 시작 기호 추가

0.  $ACC \rightarrow E$
1.  $E \rightarrow E + F$
2.  $E \rightarrow F$
3.  $F \rightarrow a$

### (2) 오토마타 구성

$$S_0 = CLOSURE(\{ [ACC \rightarrow \cdot E] \}) = \{ [ACC \rightarrow \cdot E], [E \rightarrow \cdot E + F], [E \rightarrow \cdot F], [F \rightarrow \cdot a] \}$$

$$GOTO(S_0, E) = S_1 = \{ [ACC \rightarrow E \cdot], [E \rightarrow E \cdot + F] \}$$

$$GOTO(S_0, F) = S_2 = \{ [E \rightarrow F \cdot] \}$$

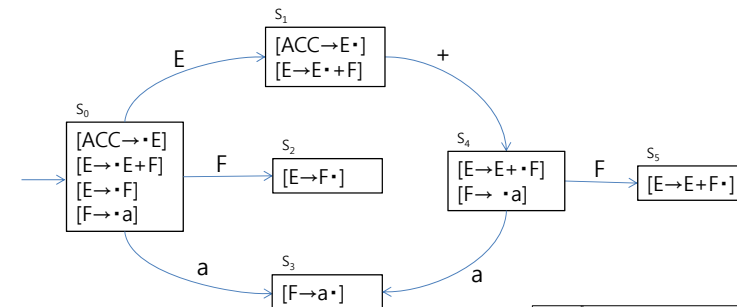
$$GOTO(S_0, a) = S_3 = \{ [F \rightarrow a \cdot] \}$$

$$GOTO(S_1, +) = S_4 = \{ [E \rightarrow E + \cdot F], [F \rightarrow \cdot a] \}$$

$$GOTO(S_4, F) = S_5 = \{ [E \rightarrow E + F \cdot] \}$$

$$GOTO(S_4, a) = S_3$$

19



$$FOLLOW(E) = \{ +, \$ \}$$

$$FOLLOW(F) = \{ +, \$ \}$$

심벌 상태	ACTION 테이블			GOTO 테이블	
	a	+	\$	E	F
0	s3			1	2
1		s4	acc		
2		r2	r2		
3		r3	r3		
4	s3				5
5		r1	r1		

20

## SLR 파싱표

- LR 파싱표를 구성하는 가장 간단한 방법
- LR(0) item 집합과 mark 심벌들에 대한 다음 상태들로 구성된 오토마타로부터 구성

- 1) 상태  $S_i$  에서 mark 심벌이 **터미널**  $a$ 인 경우  
 $GOTO(S_i, a) = S_j$  에 대해  $ACTION[i][a] = \langle \text{shift}, j \rangle$
- 2) 상태  $S_i$  에서 mark 심벌이 **논터미널**  $A$ 인 경우  
 $GOTO(S_i, A) = S_j$  에 대해  $GOTO[i][A] = j$
- 3) 상태  $S_i$  에서 **reduce item**  $[A \rightarrow \alpha \cdot]$  가 존재하는 경우  
 $FOLLOW(A)$ 에 속하는 모든 터미널  $a$ 에 대해  
 $ACTION[i][a] = \langle \text{reduce, 생성규칙 번호} \rangle$

21

## SLR 파싱표 작성 연습

Predictive 파싱에서 예로 들었던 문법

$$\begin{aligned} S &\rightarrow bAb \mid aB \mid \epsilon \\ A &\rightarrow aAb \mid bBa \\ B &\rightarrow b \mid \epsilon \end{aligned}$$

새로운 시작 기호 추가

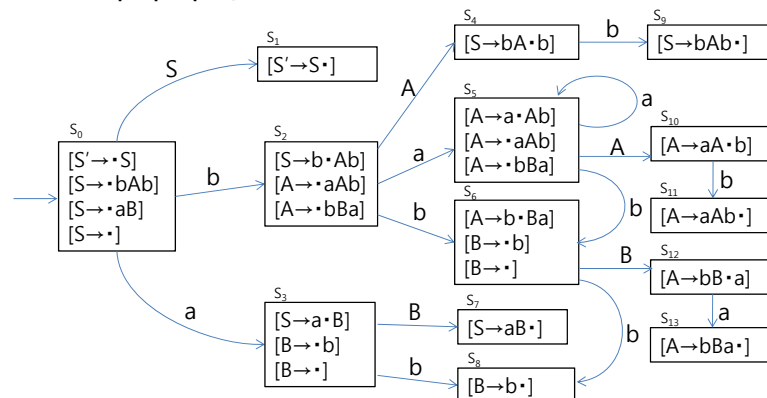
0.  $S' \rightarrow S$
1.  $S \rightarrow bAb$
2.  $S \rightarrow aB$
3.  $S \rightarrow \epsilon$
4.  $A \rightarrow aAb$
5.  $A \rightarrow bBa$
6.  $B \rightarrow b$
7.  $B \rightarrow \epsilon$

<참고> 위 문법에 대한 yacc 입력 파일을 작성하고 입력 bbab, babbabb에 대한 파싱 결과를 출력하시오.

22

## $C_0$ 와 GOTO 그래프

- 오토마타 구성



23

- FIRST와 FOLLOW

$FIRST(S) = \{ a, b, \epsilon \}$

$FIRST(A) = \{ a, b \}$

$FIRST(B) = \{ b, \epsilon \}$

$FOLLOW(S) = FOLLOW(S') = \{ \$ \}$

$FOLLOW(A) = \{ b \}$  // 책 에러 수정

$FOLLOW(B) = \{ a, \$ \}$

24

심벌 상태	ACTION 테이블			GOTO 테이블		
	a	b	\$	S	A	B
0	s3	s2	r3	1		
1			acc			
2	s5	s6			4	
3	r7	s8	r7			7
4		s9				
5	s5	s6			10	
6	r7	s8	r7			12
7			r2			
8	r6		r6			
9			r1			
10		s11				
11		r4				
12	s13					
13		r5				

<참고> 위 파싱표에 따라 bbab, babbabb에 대한 파싱 과정을 보이고 yacc로 파싱한 결과와 동일한지 비교하십시오.

25

## SLR 파싱표: 모호한 문법

$E \rightarrow E + E \mid E * E \mid a$

새로운 시작 기호 추가

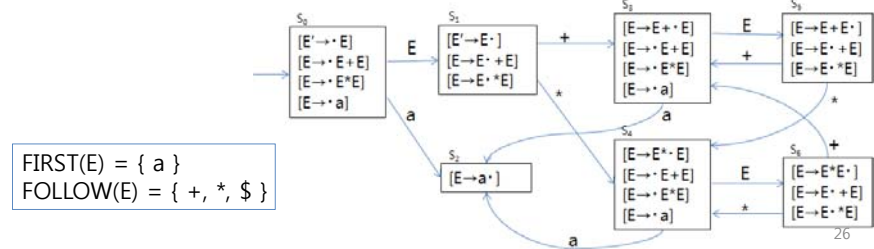
0.  $E' \rightarrow E$

1.  $E \rightarrow E + E$

2.  $E \rightarrow E * E$

3.  $E \rightarrow a$

심벌 상태	ACTION 테이블				GOTO 테이블
	a	+	*	\$	E
0					
1				acc	
2					
3					
4					
5					
6					



## LR(1) item

- LR(0) item의 reduce item  $[A \rightarrow \alpha \cdot]$ 에 대해
  - FOLLOW(A)에 속하는 모든 터미널 a에 대해 reduce 연산을 수행
  - 문제점: FOLLOW(A)에는 다른 유도 과정에서 발생하는 터미널이 포함되는 경우가 발생 가능
- LR(1) item :  $[A \rightarrow B \cdot CD, a/b/c]$ 
  - 시작 기호로부터 현재 상태까지 유도 과정을 고려
  - 시작 상태에서부터 현재 상태까지 유도 과정에서 reduce가 허용되는 터미널 집합을 lookahead라 하고, LR(0) item에 lookahead를 추가한 것
  - 추가된 생성규칙  $[S' \rightarrow \cdot S]$ 의 lookahead는 '\$' 이므로 시작 상태는  $[S' \rightarrow \cdot S, \$]$ 의 closure item에 대한 CLOSURE 집합

27

## LR(1) item의 lookahead 계산

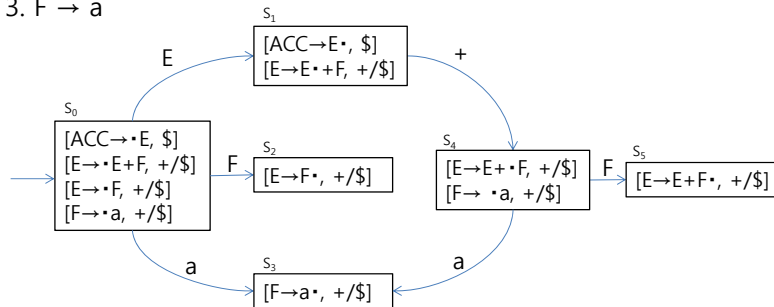
- X-transition의 경우
  - dot가 mark 기호를 넘어 다음 상태로 이동
    - Lookahead는 전 상태의 lookahead와 동일함
  - 예)  $[A \rightarrow \cdot BCD, a/b/c]$ 로부터 다음 상태 B-transition의 LR(1) item은  $[A \rightarrow B \cdot CD, a/b/c]$
- $\epsilon$ -transition의 경우
  - CLOSURE 집합을 구할 때 mark 심벌에 대한 closure item
    - $CLOSURE([A \rightarrow \cdot BCD, a/b/c]) = \{$ 
      - $[A \rightarrow \cdot BCD, a/b/c],$
      - $[B \rightarrow \cdot CD, d/e/f],$
      - $[B \rightarrow \cdot EF, d/e/f],$
      - C-생성규칙에 대한 closure item,
      - E-생성규칙에 대한 closure item  $\}$
  - B-생성규칙의 lookahead 계산:
    - $FIRST(CD\{a/b/c\}) = FIRST(C) \oplus FIRST(D) \oplus \{a, b, c\}$

28

# C<sub>1</sub>과 GOTO 그래프

## • (1) 새로운 시작 기호 추가

0.  $ACC \rightarrow E$
1.  $E \rightarrow E + F$
2.  $E \rightarrow F$
3.  $F \rightarrow a$



29

## • $S_0$ 의 LR(1) item 중에서 $[E \rightarrow \cdot E + F, +/\$]$ 의 lookahead $+/\$$ 계산

–  $[ACC \rightarrow \cdot E, \$]$ 의 mark 심벌 E에 대한  $\epsilon$ -transition에 의하여

- $[E \rightarrow \cdot E + F, \$]$ 와  $[E \rightarrow \cdot F, \$]$  추가

– 추가된 LR(1) item의 mark 심벌 E에 대해  $\epsilon$ -transition에 의하여

- $[E \rightarrow \cdot E + F, +]$ 와  $[E \rightarrow \cdot F, +]$  추가

30

## • 파싱표: SLR 파싱표와 동일함

심벌 상태	ACTION 테이블			GOTO 테이블	
	a	+	\$	E	F
0	s3			1	2
1		s4	acc		
2		r2	r2		
3		r3	r3		
4	s3				5
5		r1	r1		

31

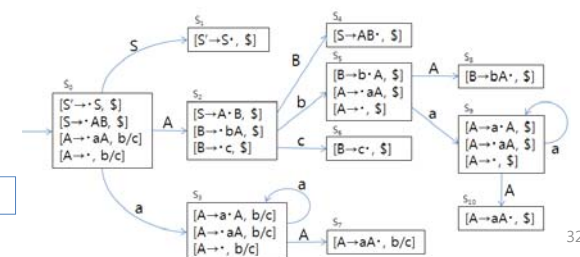
## CLR 파싱표 연습

$S \rightarrow AB$   
 $A \rightarrow aA \mid \epsilon$   
 $B \rightarrow bA \mid c$

## • (1) 새로운 시작 기호 추가

0.  $S' \rightarrow S$
1.  $S \rightarrow AB$
2.  $A \rightarrow aA$
3.  $A \rightarrow \epsilon$
4.  $B \rightarrow bA$
5.  $B \rightarrow c$

심벌 상태	ACTION 테이블				GOTO 테이블		
	a	b	c	\$	S	A	B
0	s3	r3	r3		1	2	
1				acc			
2		s5	s6				4
3	s3	r3	r3			7	
4				r1			
5	s9			r3		8	
6				r5			
7		r2	r2				
8				r4			
9	s9			r3		10	
10				r2			



aabaa에 대한 파싱

32



## LALR 파싱표

- SLR 파싱표
  - 크기 작으나 reduce 행동에서 FOLLOW에 대해 생성규칙을 적용하므로 정확도가 낮음
- CLR 파싱표
  - 정확한 lookahead에 대해 reduce
  - 파싱표의 크기가 커지는 단점
- LALR 파싱표
  - SLR과 CLR의 장단점에 대한 절충안
  - 파싱표의 크기는 SLR과 같고 lookahead는 CLR 방식에 따라 계산

33

## LALR 파싱표 구성 방법

- CLR 파싱표에서 구하는 방법
  - LR(0) item 부분이 동일하고 lookahead만 다른 상태들을 통합(merge)
- SLR 파싱 과정에 lookahead를 추가하는 방법
  - SLR 파싱 과정을 수정하여 각 상태에서 lookahead 계산 과정 추가
  - CLR 파싱표로부터 구하는 방법 보다 SLR 파싱 과정에 lookahead를 계산하는 알고리즘은 좀더 복잡

34

- CLR 파싱표: 3번과 9번, 7번과 10번 통합

심벌 상태	ACTION 테이블				GOTO 테이블		
	a	b	c	\$	S	A	B
0	s3	r3	r3		1	2	
1				acc			
2		s5	s6				4
3	s3	r3	r3	<b>r3</b>		7	
4				r1			
5	<b>s3</b>			r3		8	
6				r5			
7		r2	r2	<b>r2</b>			
8				r4			

35