Top down Parsing

2022년 5월 18일 수요일 오후 12:53

LL parsing

- Left to right scan, Left Parse
- Left most dervation 순으로 분석 -> left non terminal first
- 왼쪽부터 스캔해서 파싱하는 방식
- CFG의 일부분만 parse할수있다

Nondeterministic top down parsing

- Recursice decent with backtracking
- 규칙 선택이 잘못된 경우 bactracking 필요

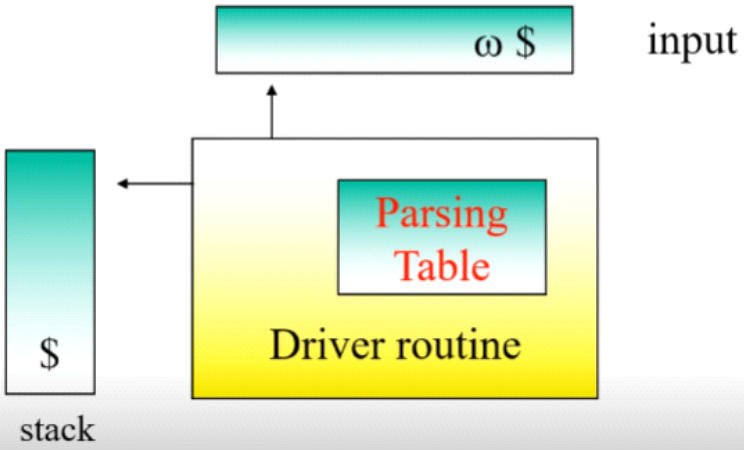
Deterministic top down parsing

- 규칙을 항상 결정적으로 선택
- Predictive parser
  - Backtracking을 배제한것이다
  - 미리 예상하고 parsing path를 정한다 -> lookahead라고 한다
  - LL(1) : one token lookahead top down predictive parser
  - Lookahead 방법
    - Parsing table을 만들어서 다음에 어떤 token이 나올지르 ㄹ찾는다
    - First(), Follow()함수를 이용해서 구현한다

Left recursion은 infinite loop 발생한다

Predictive Parser

Predictive parser의 구성



화면 캡처: 2022-05-25 오전 10:54

Turing machine, Finate state machine과 비슷한 구조이다

예) G : 1. S → aSb
2. S → bA
3. A → aA
4. A → b

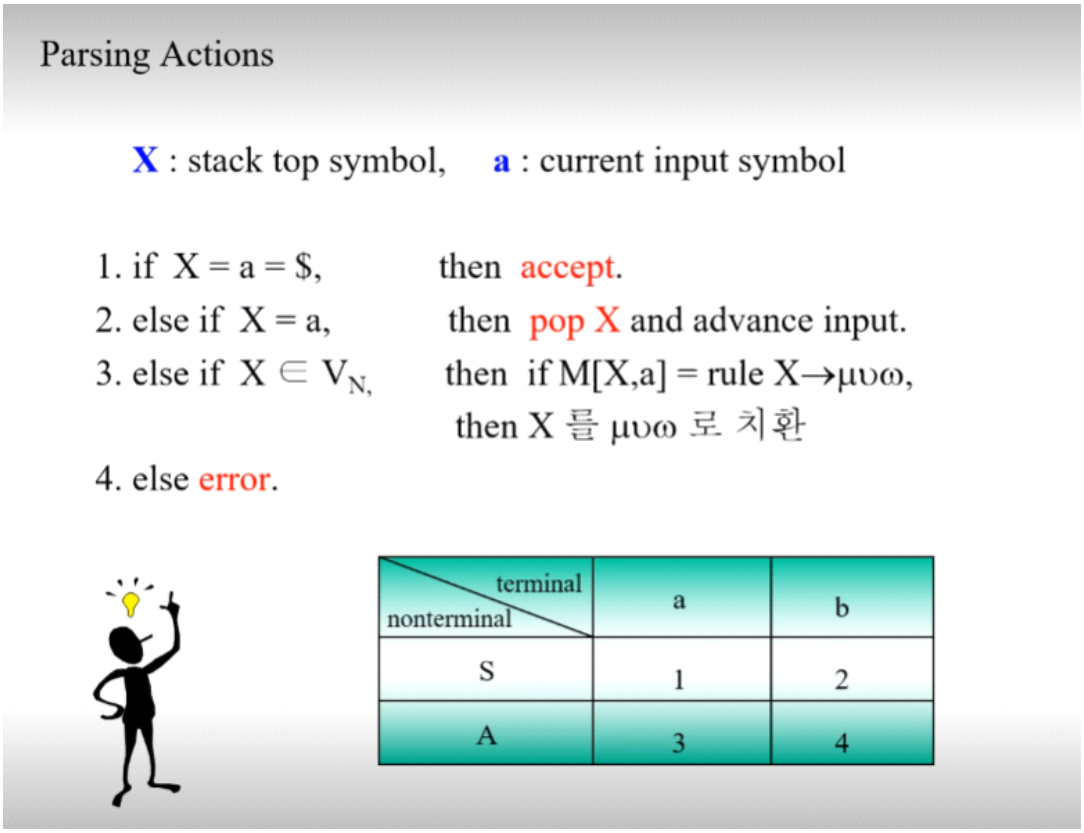
string : aabbbb

• Parsing Table:

nonterminals \ terminals	a	b
S	1	2
A	3	4

화면 캡처: 2022-05-25 오전 10:56

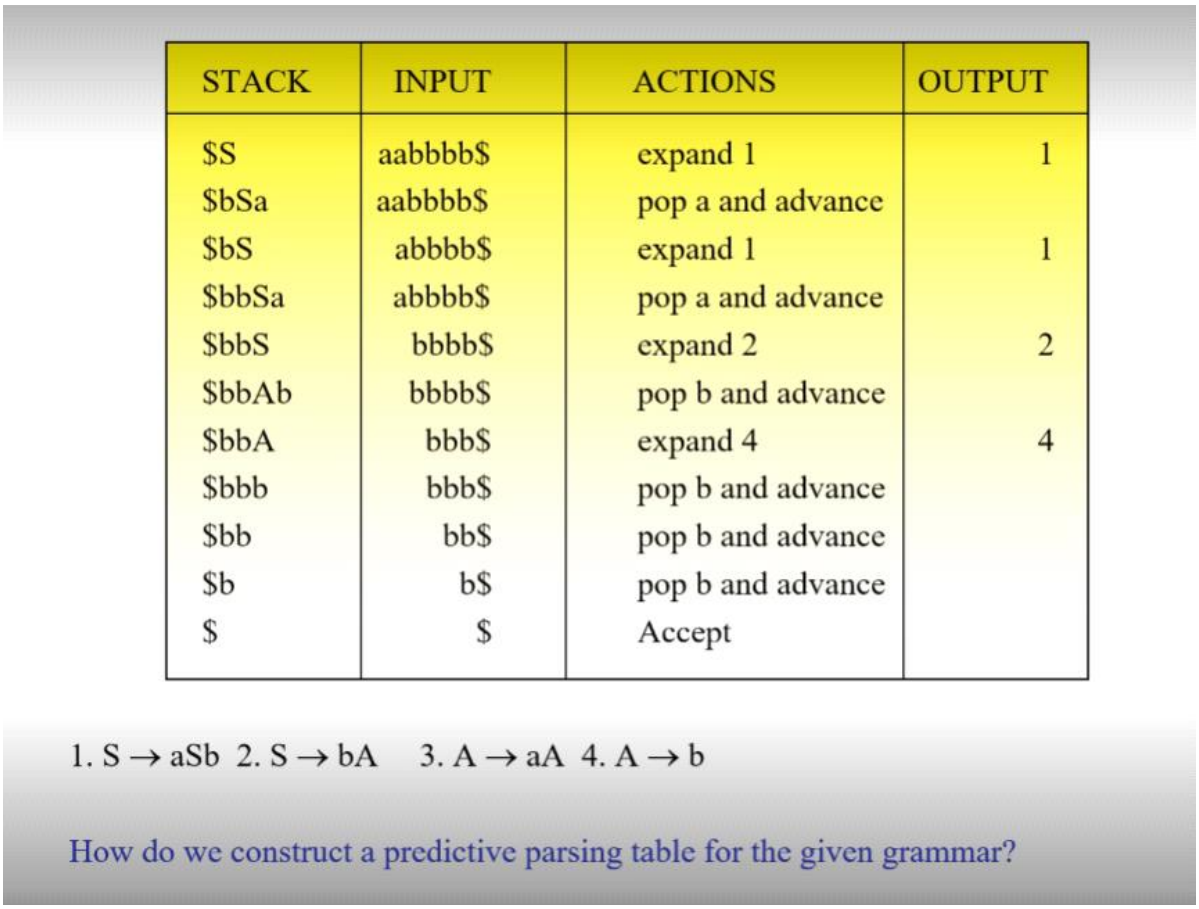
예를 들어서 스택에 S가 있고 다음 입력이 b라면 rule 2를 적용해라 지금 rule 2는 S->bA이다
이 규칙을 적용하라는 의미이다



화면 캡처: 2022-05-25 오전 10:58

If X==a==$ // $는 끝났다는 의미 이다
{
종료
}
Else if(X==a) //stack에 있는 X와 입력 a가 같다면 terminal symbol이라는 의미니까 pop해준다
{
pop X
input pointer move to next pointer
}
Else if(X∈V_n) //
{
x가 non terminal이면 parsing table에서 규칙을 찾는다
해당 규칙으로 치환한다
스택에 넣을 때 순서를 뒤집어서 push한다
abc이면 c부터 push

a
b
c
}



STACK	INPUT	ACTIONS	OUTPUT
$S	aabbbb$	expand 1	1
$bSa	aabbbb$	pop a and advance	
$bS	abbbb$	expand 1	1
$bbSa	abbbb$	pop a and advance	
$bbS	bbbb$	expand 2	2
$bbAb	bbbb$	pop b and advance	
$bbA	bbb$	expand 4	4
$bbb	bbb$	pop b and advance	
$bb	bb$	pop b and advance	
$b	b$	pop b and advance	
$	$	Accept	

화면 캡처: 2022-05-25 오전 11:04

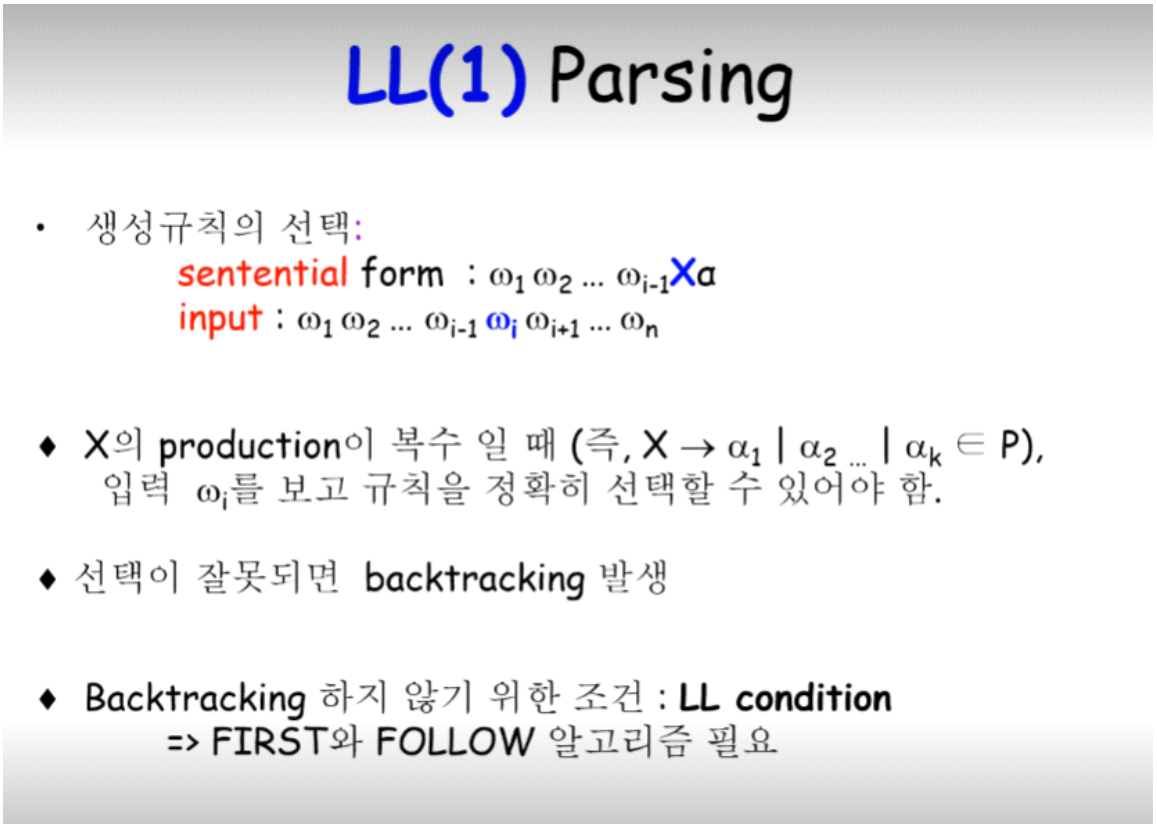
지금 stack에 start symbol이 있고 input은 aabbbbb$ 즉 a^2b^4가 유효한지 증명하는 과정이다

S와 a가 만났으니 rule 1번 적용하면 S->aSb인데 스택에는 순서가 반대서 부터 push가 되니까

2번째에 지근 $bSa인것이다 이런식으로 진행하면 $ $ Accept가 된다

이런식으로 parsing과정이 된다

그렇다면 Parsing table을 어떻게 정확하게 만들수있는가 ?
: 만들수는 있지만 모든 문법에 적용되지는 않는다
이제 그 방법에 대해 알아본다



화면 캡처: 2022-05-25 오전 11:09

모든 문법이 LL Parsing을 할 수가 없다 LL condition이 만족되어야한다

선택이 잘못되면 backtracking 발생

Bactracking 하지 않기 위한 조건

- FIRST 알고리즘
- FOLLOW 알고리즘

FIRST (X)함수

FIRST (X)

정의: FIRST(X) = { $a \in V_T \mid X \Rightarrow^* a\alpha, \alpha \in V^*$ }

즉, X가 생성하는 스트링의 시작 위치에 올 수 있는 terminal의 집합.
$X \Rightarrow^* \varepsilon$ 이면 ε도 포함.

예1)

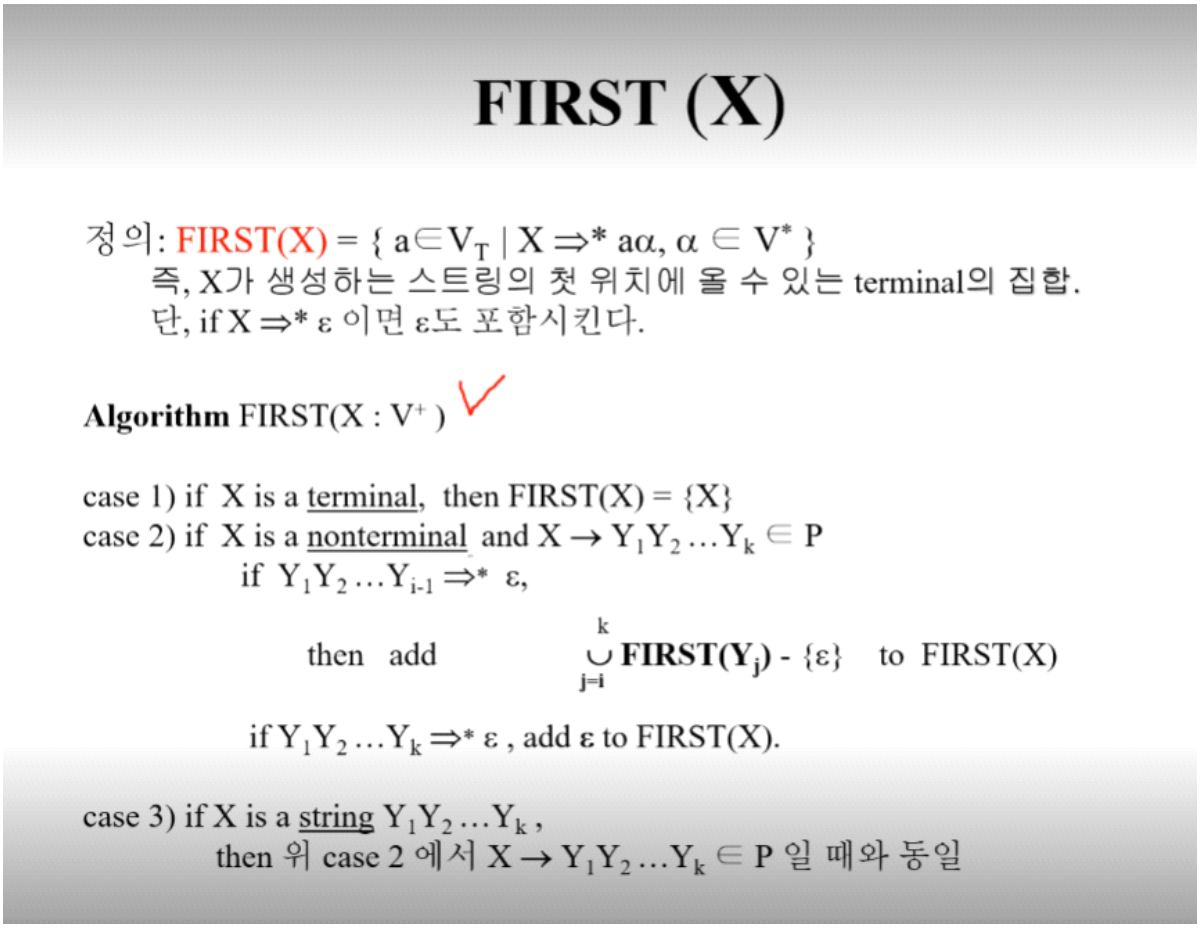
PROGRAM → begin d semi X end
X → d semi X
X → s Y
Y → semi s Y | ε

FIRST(PROGRAM) = {begin}
FIRST(X) = {d,s}
FIRST(Y) = {semi, ε}

화면 캡처: 2022-05-25 오전 11:23

X : non Terminal,Termina,String 올수있다

해당 token의 set중에서 비교대상의 이전 terminal을 찾아주는 함수이다
First(x)하면 X라고 하는 set중에서 첫번째 terminal을 찾아주는것이다.
ε 도 포함이다



화면 캡처: 2022-05-25 오전 11:24

FIRST(X)함수의 X인자에 따라서 terminal, non terminal , string인 경우를 처리하는 방법이 위에 설명 되어있다

Case 1

X가 terminal이면 frist함수에 추가

Case 2

X가 non terminal이면 그리고 X-> Y1Y2Yk 에서 첫 terminal을 first 함수 추가

만약 ε 이 있다면 first에 추가 한다

Non terminal이면 쭉 따라가서 나오는 첫번째 terminal symbol이 first 함수에 추가 된다

Case 3

X가 문자열 y1y2...라면 case2와 동일

예2) $E \rightarrow TE'$ $\quad E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$ $\quad T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid \mathbf{id}$

E, T, F, E', T에 대해 First()를 구하라

$$FIRST(E) = FIRST(T) = FIRST(F) = \{(, id\}$$
$$FIRST(E') = \{+, \varepsilon\}$$
$$FIRST(T') = \{*, \varepsilon\}$$

화면 캡처: 2022-05-25 오전 11:31

E의 First는 A-> Bβ 꼴이다 즉 T의 first는 E의 fist에 속하는 꼴이다
T의 first는 A-> Bβ 꼴이다 즉 F의 first는 T의 first에 속하는 꼴이 된다

E <- T <- F 이런 관계가 된다 즉 F의 first 모두가 E와 T에 전수 된다

F의 first 는 {(,id}
E'의 first는 {+,ε }
T'의 first는 {*,ε }

보충 설명
위의 예제로 설명 이어간다

First E 는 T를 만남 그럼 First T와 같은거임 First T는 F를 만남 그럼 First F와 같은거임

First F는 (와 id가 있다 그럼 First E ,T ,F는 {(,id}가 된다

FOLLOW(X:V_N)

비교대상 바로 다음에 나올수있는 terminal들의 집합을 구해주는 함수이다

Follow(A)라고 하면 A라는 Symbol뒤에 나올수있는 symbol들을 포함하는 것이다

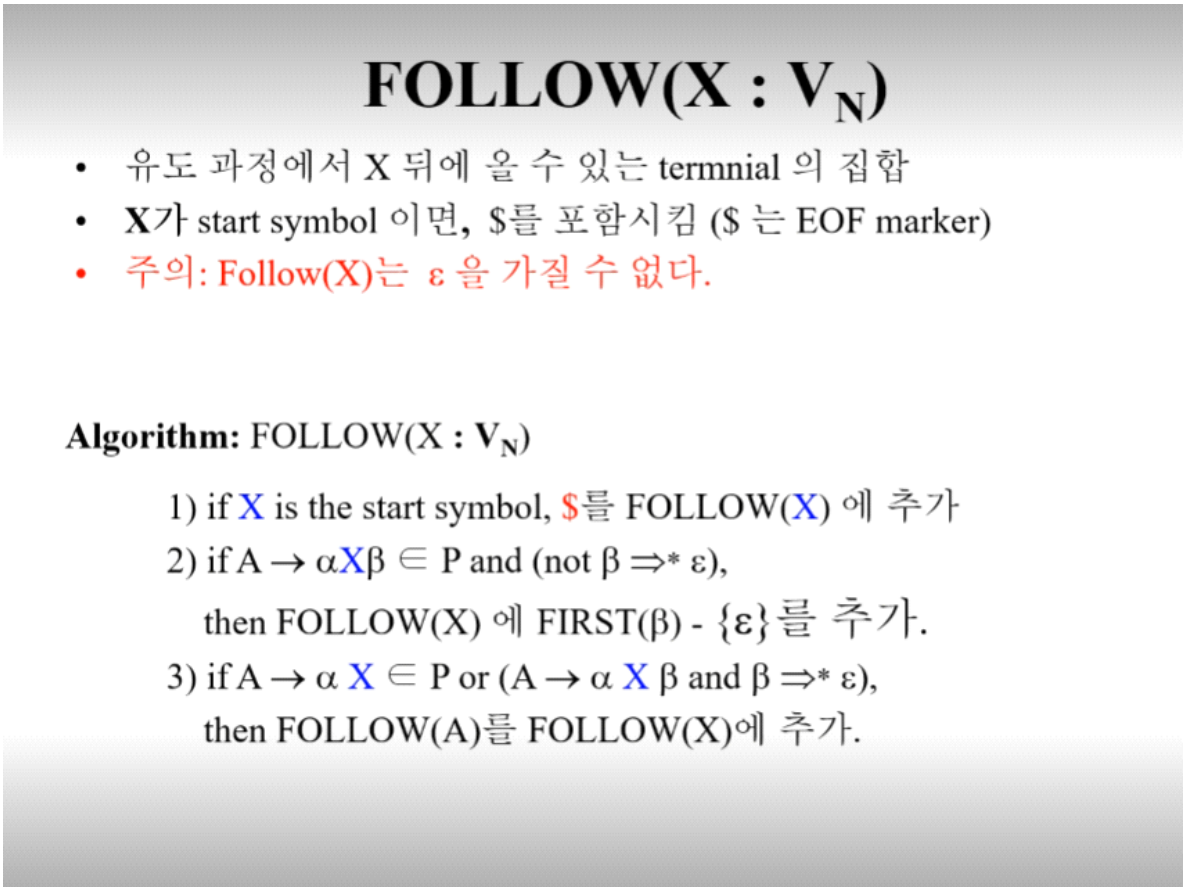
Non terminal symbol의 바로 다음 terminal symbol이 follow가 된다

αAaB에서 A의 바로 다음 terminal symbol이 a이므로, aAB의 꼴에서 FOLLOW(A)를 찾기 위해서는 B의 FIRST를 찾으면 된다.

예시

X -> YAaZ

Follow(A) = {a}



화면 캡처: 2022-05-25 오전 11:35

Start symbol의 follow는 $를 반드시 포함한다

Floow (X)는 ε 을 가질수없다
여기서 X parameter는 non terminal이다

1. X가 start symbol이면 $를 FOLLOW(X)에 추가
2. A->αXβ 그리고 b가 ε 을 생성하지 않는 경우
    a. FOLLOW(X)에 FIRST(β) - {ε}을 빼준 결과를 추가한다 왜냐하면 follow는 ε이 올수없으니까
    b. X뒤에 무언가가 있다는 의미이다
3. A - >α X 또는 A->αXβ β => ε 생성 가능 한 경우
    a. Follow A를 Follow X에 추가 한다

보충 설명

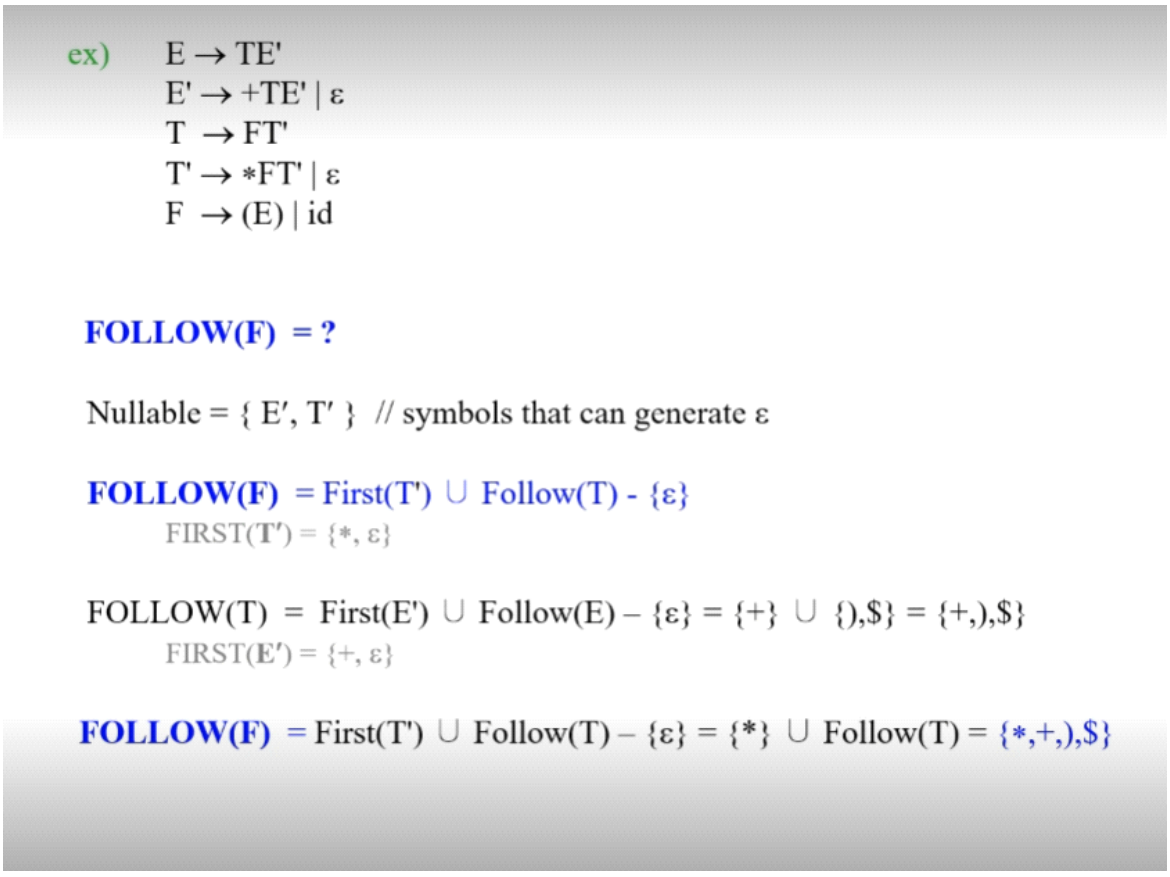
위의 규칙을 잘 생각해보면 아래와 같은 규칙으로 정리 할 수 있다

Follow(X) :

a. Y -> αXβ => First (β) - ε
b. Y -> αXβ && ε ∈ First(β) [First β가 ε 생성한다면] => Follow(Y)이다
c. Y-> αX => Follow(Y)

기본 설명 다시 리마인드 해보면 대상 X 우변에 터미널 집합이다

그니까 X를 찾고 우변의 터미널 심볼을 찾으면 된다



화면 캡처: 2022-05-25 오전 11:49

보충설명에서 정리한 규칙으로 Follow적용 시켜 본다

Follow E = {),$} E는 start symbol이기 때문이다

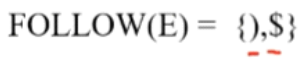
Follow T = T 뒤에 E'이 온다 E'은 ε 을 생성할수 있는 production이니까 규칙 2번에 해당한다 즉 Follow E'이 추가 됨

= Follow E' U First E'[1번 규칙] 이 온다

Follow E' = E'뒤에 오는거 없다 3번 규칙에 의해 Follow E' U Follow E

Follow T' = T'뒤에 오는거 없다 3번 규칙에 의해 Follow T' U Follow T

Follow F = F뒤에 T'오니까 규칙 1번 First T' , T'은 ε 생성 가능하니까 ' ,Follow T'



화면 캡처: 2022-05-25 오전 11:49

FOLLOW(E)의 의미 : E뒤에 올수있는 terminal 집합 , 우변에 E가 있는곳을 찾아본다
E는 start symbol이다

FOLLOW(F) 는 위의 결과가 나온다 ε 빼준다

Predictive 파싱 테이블 구현

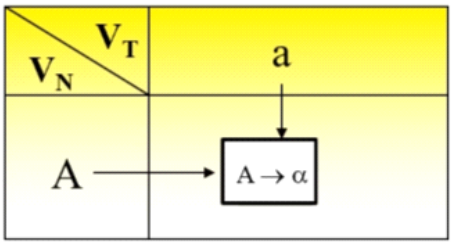
Predictive 파싱 테이블의 구현

- Stack top과 next input이 각각 A, a 일 경우를 생각해보자.

만일 $A \rightarrow \alpha \in \mathbf{P}$ 이고 $a \in FIRST(\alpha)$ 이면 $A \rightarrow \alpha$ 를 적용한다.

Otherwise, $\alpha \Rightarrow^{*} \varepsilon$ 이고 $a \in FOLLOW(A)$ 이면 $A \rightarrow \alpha$ 를 적용한다.

- LL(1) parsing table:



M[A,a] = n 이면 규칙 n을 이용하여 A를 확장한다.

M[A,a] 가 blank 이면 syntax error

화면 캡처: 2022-05-25 오전 11:52

Algorithm : Predictive parser의 parsing table 구현

for 문법의 모든 규칙 $A \rightarrow \alpha$에 대하여 다음 작업 수행
  for 모든 입력 기호 a에 대해 다음 작업 수행

  if $a \in FIRST(\alpha)$
    M[A,a] 에 $A \rightarrow \alpha$를 추가

  else if $\alpha \Rightarrow^{*} \varepsilon$ then
    for FOLLOW(A)에 속한 모든 입력기호 b에 대해,
      M[A,b] 에 $A \rightarrow \alpha$를 추가

화면 캡처: 2022-05-25 오전 11:57

Algorithm : Predictive parser의 parsing table 구현

For each production A→α,

1. ∀a ∈ FIRST(α), M[A,a] := <A→α>
2. if α ⇒* ε, then ∀b ∈ FOLLOW(A), M[A,b] := <A→α>.

ex) G: 1. E → TE' 2. E' → +TE' 3. E' → ε 4. T → FT'
5. T' → *FT' 6. T' → ε 7. F → (E) 8. F → id

FIRST(E)=FIRST(T)=FIRST(F)={ (, id }
FIRST(E')={ + , ε } FIRST(T')={ * , ε }
FOLLOW(E) = FOLLOW(E') = {) , $ }
FOLLOW(T) = FOLLOW(T') = { + ,) , $ }
FOLLOW(F) = { + , * ,) , $ }

화면 캡처: 2022-05-25 오전 11:59

테이블 생성 규칙

1. A-> α 에서 First α 가 Terminal이면 테이블에 해당 규칙과 함께 매핑 하면됨
2. 만약 α 가 입실론을 생성한다면 Follow A의 결과를 규칙과 함께 매핑 하면 된다

자 목표는 규칙을 parsing table로 만드는 과정을 살펴 보는것이다

1-8번까지의 규칙을 parsing table로 만드는 과정을 어떻게 하냐면 non T에 대해 FIRST(α)를 추가를 해주어야한다

1번 부터 보면 E-> TE' A : E, α : T

E에 대해서 FIRST(T)의 결과 { (, id }를 parsing table에 추가 할 수 있다

그래서 전체적인 테이블이 만들어지만 아래와 같다

ex) G: 1. E → TE' 2. E' → +TE' 3. E' → ε 4. T → FT'
5. T' → *FT' 6. T' → ε 7. F → (E) 8. F → id

FIRST(E)=FIRST(T)=FIRST(F)={ (, id }
FIRST(E')={ + , ε } FIRST(T')={ * , ε }
FOLLOW(E) = FOLLOW(E') = {) , $ }
FOLLOW(T) = FOLLOW(T') = { + ,) , $ }
FOLLOW(F) = { + , * ,) , $ }

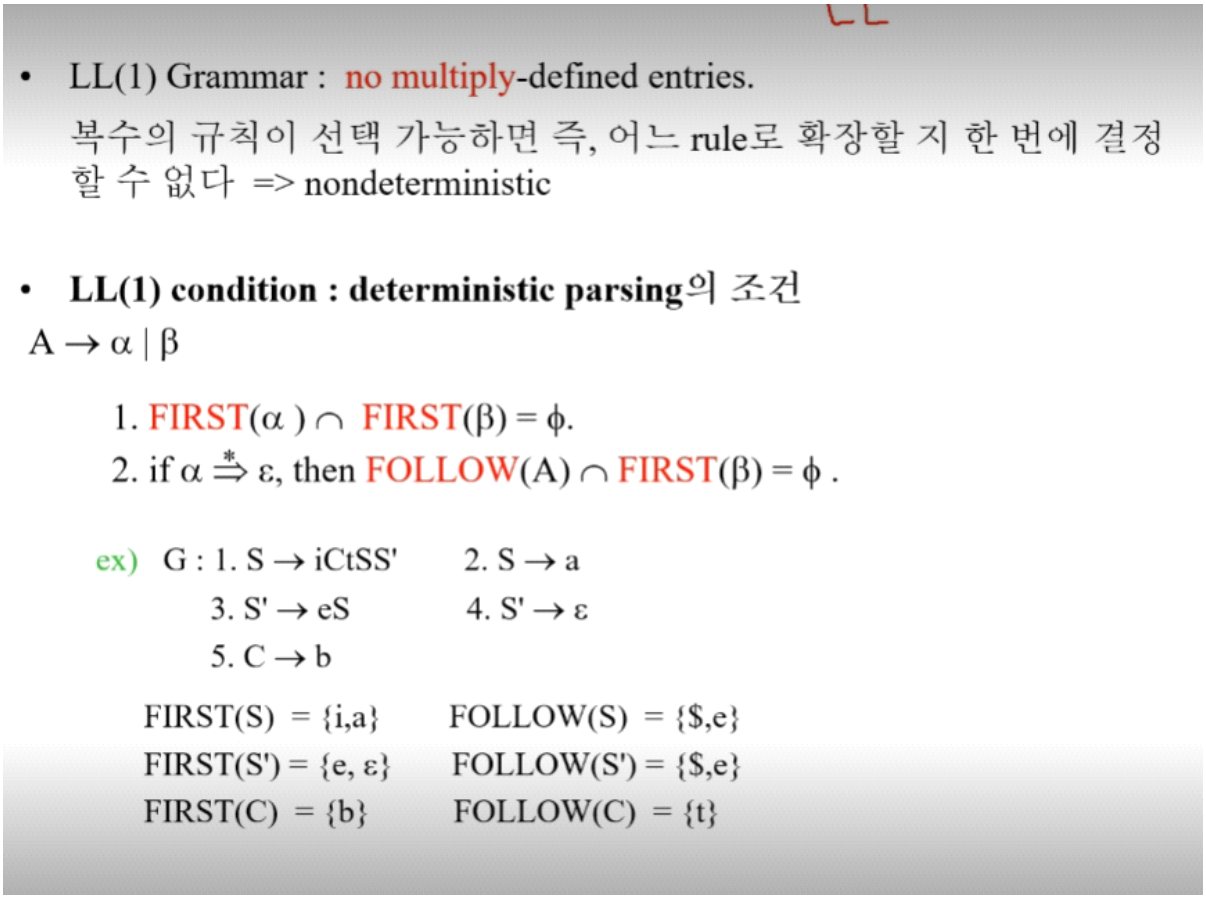
Nonterminal \ Terminal	id	+	*	(	)	$
E	1			1		
E'		2			3	3
T	4			4		
T'		6	5		6	6
F	8			7		

화면 캡처: 2022-05-25 오후 12:04

E이고 id 또는 (를 생성한다면 1번 규칙 적용해라는 것을 테이블 보고 알수있다

E'은 ε 생성하니까 Follow E'을 테이블 규칙에 넣는다

T' -> ε 이니까 FOLLOW(T')



화면 캡처: 2022-05-25 오후 12:09

LL condition의 조건을 위에서 확인할 수 있다

LL(x) : x는 lookahead의 개수이다
LL(1)은 lookahead를 1개만 한다는 의미이다

테이블 한칸에 규칙이 두개 이상 있으면 backtracking일어난다는 의미 이니까 최대 한 개만 있어야한다

A-> α | β

1. First α n First β = 공집합
2. If 알파 가 입실론을 생성하면 Follow A n First β = 공집합

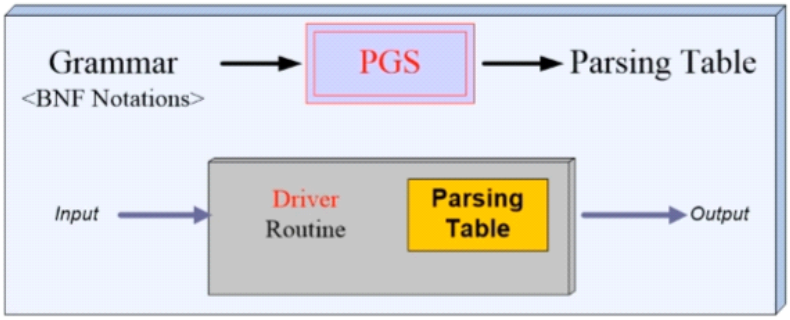
예제

S - > iCtSS' | a
S' -> eS | ε

First S는 i 또는 a이다 겹치지 않음 공집합이니까 만족

α 가 입실론을 생성하지 않기 때문에 2번째도 만족

Parser Generator 구현



- Driver routine 은 parsing table을 구동하는 부분으로 문법에 독립적으로 일정
- Parsing table은 Grammar에 따라 차이가 있으나 gramma로부터 생성 가능
- 생성된 Parsing table과 driver routine을 결합하면 최종적인 parser가 완성됨

화면 캡처: 2022-05-25 오후 12:14