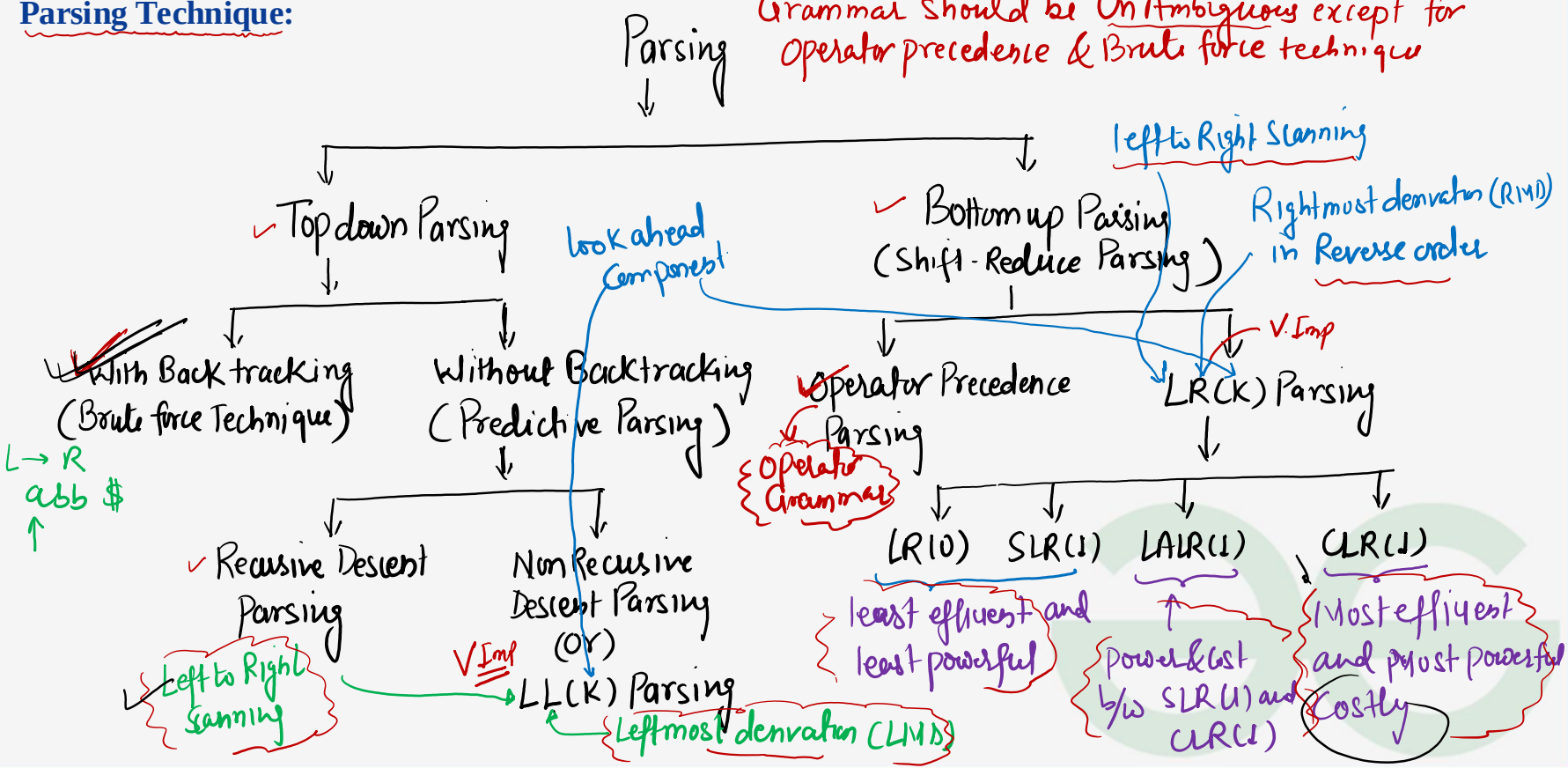
Compiler Design

Today's Class Topics

-Parsing Technique
-Top-down Parsing
-First Function
-Follow Function
-LL(k) Parser and its necessary condition
-How to construct LL(1) Parsing Table
-How to Test LL(1) Grammar etc.

Compiler Design

Parsing Technique:



Compiler Design

States: n_1 n_2 n_3 n_4

Note:

1) $n_1 = n_2 = n_3 \leq n_4$

2) No. of Shift entries of in $LR(0) = SLR(1) = LALR(1) \leq CLR(1)$

3) No of Reduce entries $LR(0) \geq SLR(1) \geq LALR(1) \geq CLR(1)$

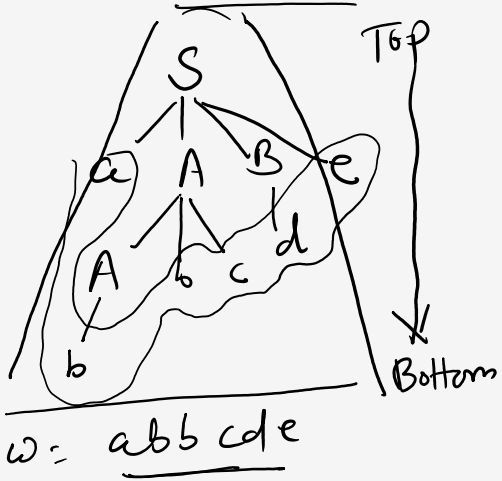
4) Power of $LR(0) < SLR(1) < LALR(1) < CLR(1)$

Compiler Design

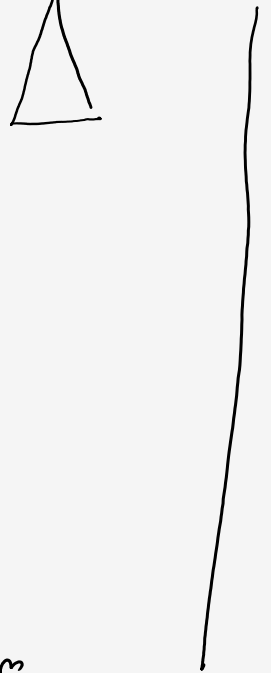
Top down Passing (LMD)

G: S → aABe
A → Abc | b
B → d

w = abbcde

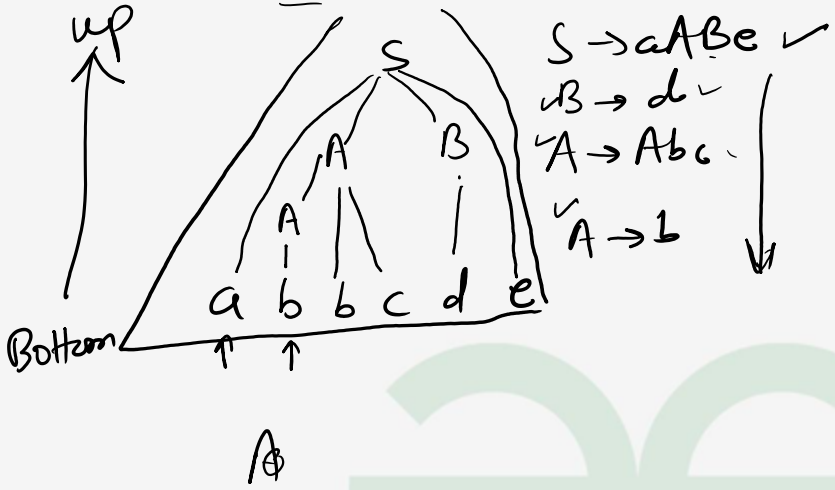


w = abbcde



Bottom up passing (RMD in Reverse order)

G: S → aABe
A → Abc | b
B → d



Compiler Design

Brute Force Technique: ✗

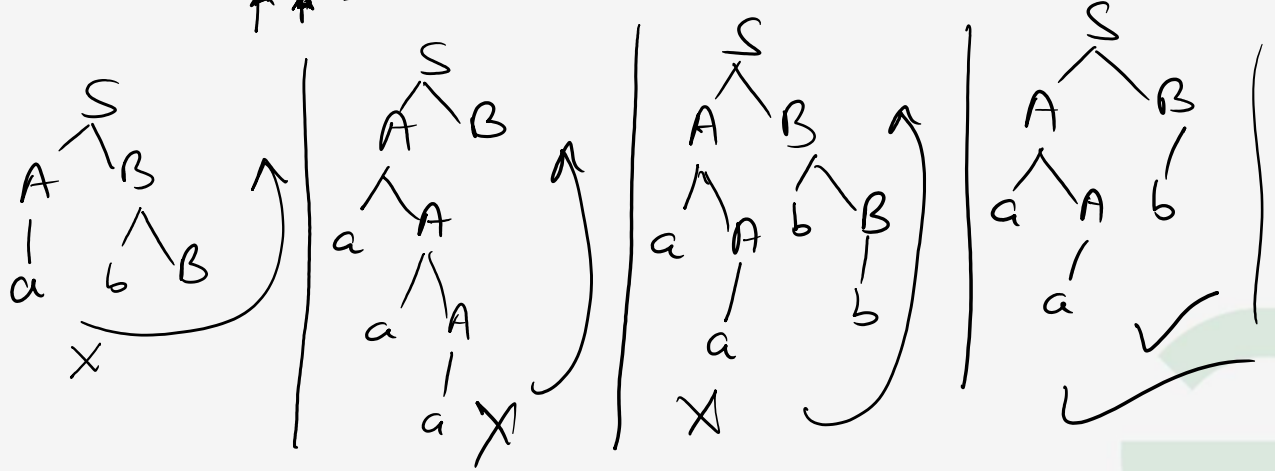
10000

100000

$G = \{S \to AB,\ A \to aA \mid a,\ B \to bB \mid b\}$

$\omega = aaab\$$

aab



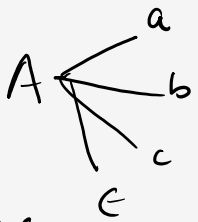
Compiler Design

Compiler Design

First and Follow Function:

First Function: First(A) is the set of terminals that are the first symbols (Terminal Symbol) on the right side of the arrow in every production of A

$A \rightarrow a\_ \mid b\_\_ \mid c\_\_ \mid \epsilon$



First(A) = {a, b, c, ϵ}

Ex: $A \rightarrow aA \mid bb \mid cB$

First(A) = {a, b, c}

Ex: $A \rightarrow aA \mid \epsilon$

First(A) = {a, ϵ}

$A \rightarrow aA \mid BC$

First(A) = {a, First(BC)}

First(A) = First(B) − ϵ

Ex: $A \rightarrow BCD$

$B \rightarrow bB \mid \epsilon$

$C \rightarrow c \mid \epsilon$

$D \rightarrow d \mid \epsilon$

First(A) = {b, c, d, ϵ}

First(B) = {b, ϵ}

First(C) = {c, ϵ}

First(D) = {d, ϵ}

First(A) : First(BCD) = First(B)

= {b}

Compiler Design

Procedure:

1. First(a) = {a}
2. if $A \to a \Rightarrow First(A) = \{a\}$
3. if $A \to \epsilon \Rightarrow First(A) = \{\epsilon\}$
4. if $A \to BC \Rightarrow First(A) = First(B)$ iff First(B) does not Contain ϵ
5. if $A \to BC \Rightarrow First(A) = (First(B) - \{\epsilon\}) \cup First(C)$

Ex: G: $S \to ABC$, $A \to a$, $B \to bB \mid a$, $C \to \epsilon$

First(S) = First(ABC) = First(A) = {a}
First(A) = {a}
First(B) = {a, b}
First(C) = {ε}

First(S) = First(ABC)
= {a, ✗} ∪ {a, b}
= {a, b}

Ex:
G: $S \to ABC$ First(S) = {a, b}
$A \to aA \mid \epsilon$ First(A) = {a, ε}
$B \to bB \mid a$ First(B) = {b, a}
$C \to cC \mid a$ First(C) = {c, a}

6. $A \to BCD$
First(A) = (First(B) − {ε}) ∪ (First(C) − {ε}) ∪ First(D)

Compiler Design

Ex: $S \to A\,b\,(C)$ — First(S) = {a, b, c, ∈} ✓

$A \to aA \mid B$ — First(A) = {a, b, ∈} ✓

$B \to bB \mid \in$ — First(B) = {b, ∈} ✓

$C \to cC \mid b \mid \in$ — First(C) = {b, c, ∈} ✓

$A \to Ae$

$A \to \in \cdot a$

$A \to c$

$B \to Ca$

$B \to a$

$B \to C$ ✗

$B \to Ca$

Ex: $S \to aA \mid BC$, First(S) = {a, b, c}

$A \to Aa \mid \in$ — First(A) = {∈, a}

$B \to Ca \mid bB$ — First(B) = {b, c, a}

$C \to cA \mid \in$ — First(C) = {c, ∈}

Ans

First(S) = First(aA) ∪ First(BC)

= {a} ∪ {a, b, c}

= {a, b, c}

First(B) = First(Ca) ∪ First(bB)

= First(c, a) ∪ {b}

Compiler Design

Ex:

$S \to ACB \mid CbA \mid Ba$

$A \to da \mid BC$

$B \to g \mid \epsilon$

$C \to h \mid \epsilon$

$first(S) = \{d, g, h, \epsilon, b, a\}$

$first(A) = \{d, g, h, \epsilon\}$

$first(B) = \{g, \epsilon\}$

$first(C) = \{h, \epsilon\}$

Ex:

$S \to AaB \mid bC$

$A \to BC \mid bA$

$B \to aB \mid \epsilon$

$C \to SA \mid a$

$first(S) = \{a, b\}$

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

$first(B) = \{a, \epsilon\}$

$first(C) = \{a, first(S)\}$

$= \{a, b\}$

$first(S) = first(AaB) \cup first(bC)$

$first(A) = first(BC) \cup first(bA)$

$= \{b, a, first(C)\}$

$first(C) = first(SA) \cup first(a)$

$= \{a, first(S)\}$ — (1)

$first(S) = \{b, first(A)\}$

$= \{b, b, a, first(C)\}$ — (2)

$= \{a, b, first(C)\}$

$first(C) = \{a, b, first(C)\}$

$\{a, b, first(C)\}$'s

$\{a, b\}$ - fin

Compiler Design

G: $E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

id

$First(E) = First(T) = First(F) = \{(, id\}$

$First(T) = \{(, id\}$

$First(F) = \{(, id\}$

$E \to TE'$

$E' \to +TE' \mid \in$, $First(E') = \{+, \in\}$

$T \to FT'$

$T' \to *FT' \mid \in$, $First(T') = \{*, \in\}$

$F \to (E) \mid id$

$First(E) = First(T) = First(F) = \{(, id\}$

Compiler Design

Compiler Design

Follow Function: Follow(A) is the set of terminals that present immediately to the right side of A wherever A is on the RHS of the arrow → A ( )

$ S $

Procedure:

1- follow(S) = {$} if S is the start symbol

2- if A → αBβ ⇒ follow(B) = first(β) iff First(β) does not contain ∈

3- if A → αBβ ⇒ follow(B) = (first(β) − {∈}) ∪ follow(A) , if β contain ∈

4- if A → αB. ⇒ follow(B) = follow(A)

first(β) = {a ∈}

A $)

αB. $

follow(B) = first(β) =

A → αBa

{c,

Compiler Design

$G = \{ S \to AaAb \mid BbBa,\ A \to cA \mid a,\ B \to bB \mid b \}$

follow(S) = {$}

follow(A) = {a, b}

follow(B) = {a, b}

S →	AaAb | BbBa
A →	cA | a
B →	bB | b

$B \to bB \mid b$

$S \Rightarrow BbBa$

$A \to cA$

$B \to bB$

Compiler Design

Ex: $S \to AaBb \mid AC$

$A \to aB \mid b$

$B \to a$

$C \to c$

$S \to A$

$A \to cA$ follow(A) = follow(A)

follow(S) = {$}

follow(A) = {a, first(C)} = {a, c}

follow(B) = {b, follow(A)} = {a, b, c}

follow(C) = {follow(S)} = {$}

Ex:

$S \to ABC$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

$C \to cC \mid \epsilon$

follow(S) = {$} = {$}

follow(A) = {b, c, follow(S)} = {b, c, $}

follow(B) = {c, follow(S)} = {c, $}

follow(C) = {follow(S)} = {$}

first(B) = {b, ε}

first(C) = {c, ε}

first(A) = {a, ε}

Compiler Design

$G = \{ S \xrightarrow{S \to S} SAB \mid \epsilon,\ A \to AB \mid \epsilon,\ B \to BC \mid Ca,\ C \to cCb \mid a \}$

$S \to SAB \mid \epsilon$

$A \to AB \mid \epsilon$

$B \to BC \mid Ca$

$C \to cCb \mid a$

$first(S) = \{\epsilon, a, c\}$

$first(A) = \{\epsilon, a, c\}$

$first(B) = \{a, c\}$

$first(C) = \{a, c\}$

$follow(S) = \{\$, first(AB)\} = \{\$, a, c\}$

$follow(A) = \{first(B)\} = \{a, c\}$

$follow(B) = \{first(C), follow(A), follow(S)\}$

$follow(C) = \{b, follow(B), a\}$

$follow(B) = \{a, c, \$\}$

$follow(C) = \{a, b, c, \$\}$

Compiler Design

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E) \mid id$

$First(E) = First(T) = First(F) = \{(, id\}$

$First(E') = \{+, \epsilon\}$

$First(T') = \{*, \epsilon\}$

$Follow(T') = Follow(T)$
$= \{+,), \$\}$

$Follow(E) = \{\$,)\}$

$Follow(E') = \{Follow(E)\} = \{\$,)\}$

$Follow(T) = \{+, Follow(E)\} = \{+,), \$\}$

$Follow(T') = \{Follow(T)\} = \{+,), \$\}$

$Follow(F) = \{*, Follow(T)\} = \{*, +,), \$\}$

Ans

$T \rightarrow FT'$

$T \rightarrow F(*)$

$T \rightarrow F$

Compiler Design

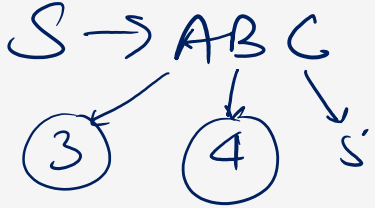
Compiler Design

Compiler Design

Compiler Design

Q.1: Consider the following Production $S \rightarrow ABC$, of the no. of elements in First(A), First(B), and First(C) are 3,4,5 elements respectively and all these First sets contain 'ϵ' and the remaining all symbols are different. Then find no. of elements in First(S)? — ?

Soln:



$First(S) = \{First(A) - \epsilon\} \cup \{First(B) - \epsilon\} \cup \{First(C)\}$

$= 2 + 3 + 5 \Rightarrow 10$

Compiler Design

Question: Consider the following Production $S \rightarrow ABC$, if the no. of elements in First(A), First(B) and First(C) are 3, 4, 5 elements respectively and all these First sets Contains $\{a, \epsilon\}$ and the remaining all symbols are different. Then find no. of elements in First(S)?

Soln:

$S \rightarrow A\ B\ C$

3 4 5

4_{-3}

First(A) = {a, ∈, ---- }

First(B) = {a, ∈, ___ }

First(C) = {a, ∈, ___ }

5 minute Break

$$First(S) = (First(A) - \epsilon) \cup (First(B) - \epsilon) \cup First(C)$$

$$= 2 + 2 + 4 \Rightarrow 8$$

Compiler Design

Top down parser:

LL(1) Parser

- LL(1) Grammar
- LL(1) Table Construction
- LL(1) Parsing
- LL(1) Grammar Testing

Left to Right Scanning → LL(K) ← Leftmost derivation

LookAhead Component

$x \to y$

if x Then y

LL(0) LL(1) LL(2) ... LL(K)

Predictive Parsing

LL(1): It is a CFG ✓

1. It is UnAmbiguous ✓
2. It is non left recursive ✓
3. It is Left factored (Deterministic CFG) ✓

a a s

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \epsilon$

$x \to y \;\&\; vy \to vx$

$x \to y \neq y \to x$

(Left Recursive ∨ left factor ∨ Ambiguous) → LL(1)

LL(1) → (Left Recursive ∨ Left factor ∨ Ambiguous)'

LL(1) → (Non left Recursive ∧ Left factored ∧ UnAmbiguous)

(Non left Recursive ∧ Left factored ∧ UnAmbiguous) → LL(1)

Compiler Design

CFG

CFG to LL(1) Grammar:

Step1: Take any CFG

↓ Remove Ambiguity → No Algo. Exist (Undecidable)

Step2: Convert to UnAmbiguous CFG

↓ Elimination of Left Recursion (Algo. Exist)

Step3: Non Left Recursive CFG

↓ Left factoring Algo

Step4: Left factored CFG

↓

Step5: LL(1) Grammar

Compiler Design

$B \to bB \mid \epsilon$, $First(B) = \{b, \epsilon\}$

Construction of LL(1) Parsing Table:

We consider every Production of the form $A \to \alpha$ and Proceed as follows

i) We write $A \to \alpha$ in $M[A, x]$ where $x \in First(\alpha)$

ii) if $First(\alpha) = \{\epsilon\}$ then Add $A \to \alpha$ in $M[A, y]$ where $y \in follow(A)$

iii) The Unfilled entries will be Considered as Syntax error

iv) if all the entries in the table are single then the Grammar is LL(1). Otherwise not LL(1)

$B \to \epsilon$

$M[B, follow(B)]$

$B \to \epsilon$

Size of Table $= |V| * (|T| + 1)$

Ex: Construct LL(1) Parsing Table.

$G = \{ S \to AB,\ A \to aA \mid b,\ B \to bB \mid \epsilon \}$

$first(S) = \{a, b\}$

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

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

$follow(S) = \{\$\}$

$follow(A) = \{b, \$\}$

$follow(B) = \{\$\}$

$M[S, x]$ $x \in first(AB)$

$S \to AB$

$V = 3$

$T = 2$

$|V| * (|T| + 1)$

$3 * 2 \to 9$

M

	a	b	$
S	S→AB	S→AB	error
A	A→aA	A→b	error
B	error	B→bB	B→ε

G:

$E \to TE'$

$E' \to +TE' \mid \epsilon$

$T \to FT'$

$T' \to *FT' \mid \epsilon$

$F \to (E) \mid id$

First(E) = {(, id}

First(E') = {+, ε}

First(T) = {(, id}

First(T') = {*, ε}

First(F) = {(, id}

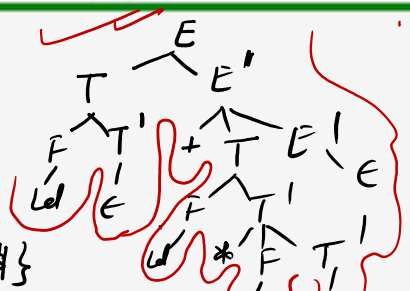
Follow(E) = {), $}

Follow(E') = {), $}

Follow(T) = {+,), $}

Follow(T') = {+,), $}

Follow(F) = {+, *,), $}



int a, b, c;

int a, b, c

id + id * id

LL(1) Parsing Table (M)

	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to id$			$F \to (E)$		

id + * id

ω = id + id * id

STACK	Input	output
$E	id + id * id $	$E \to TE'$
$E'T	id + id * id $	$T \to FT'$
$E'T'F	id + id * id $	$F \to id$
$E'T' id	id + id * id $	$T' \to \epsilon$
$E'	+ id * id $	$E' \to +TE'$
$E'T+	+ id * id $	$T \to FT'$
$E'T	id * id $	$F \to id$
$E'T'F	id * id $	$T' \to *FT'$
$E'T' id	id * id $	$F \to id$
$E'T'	* id $	$T' \to \epsilon$
$E'T'F*	* id $	$E' \to \epsilon$
$E'T'F	id $	
$E'T' id	id $	
$E'	$	
$	$	