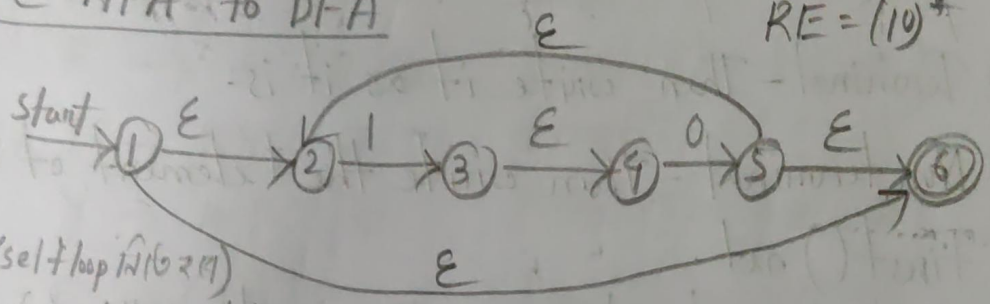
FT (L-4)

ε-NFA to DFA

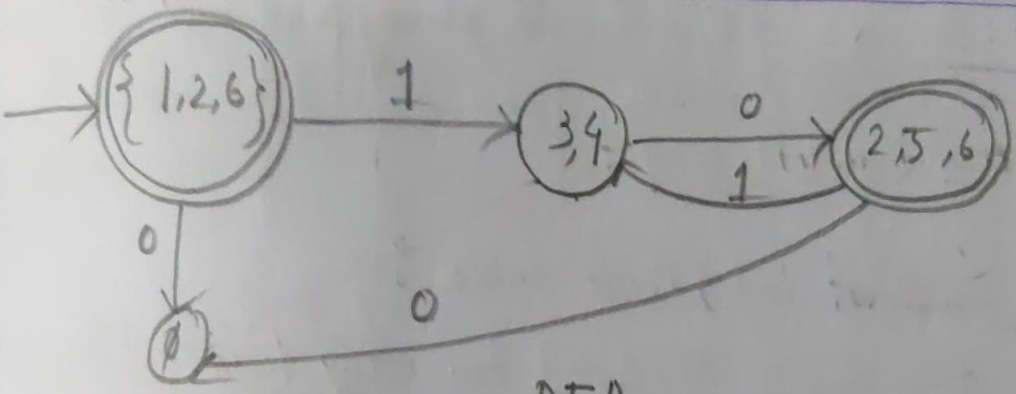
RE = (10)*



(self loop in 6 & 1)

ε-closure (1) = {1, 2, 6}
ε-closure (2) = {2}
ε-closure (3) = {3, 4}
ε-closure (4) = {4}
ε-closure (5) = {2, 5, 6}
ε-closure (6) = {6}

	0	1
→ * {1, 2, 6}	∅	{3, 4}
{3, 4}	{2, 5, 6}	∅
* {2, 5, 6}	∅	{3, 4}



DFA

* First and Follow

$E \rightarrow TE'$ — First(E) = { (, id }

$E' \rightarrow +TE' \mid \varepsilon$ — First(E') = { +, ε }

$T \rightarrow FT'$ — First(T) = { (, id }

$T' \rightarrow *FT' \mid \varepsilon$ — First(T') = { *, ε }

$F \rightarrow (E) \mid id$ — First(F) = { (, id }

Follow(E) = {), $ }

Follow(E') = {), $ }

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

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

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

*Saturday Quiz 7pm
7:30

→ One lookahead token

LL(1) - Parser { Algorithm for parsing }
↓ → Left to right derivation
Input is read from left to right

* How to construct LL(1) Parse table :

(I) Add $A \to \alpha$ in $M[A, a]$ for every terminal 'a' in First(α)

(II) If First(α) contains ε then add $A \to \alpha$ in $M[A, b]$ for every terminal 'b' in Follow(A)

e.g: $S \to aA \mid bB$
$A \to aA \mid b$
$B \to bA \mid c \mid \varepsilon$

	First	Follow
S	{a, b}	
A	{a, b}	
B	{b, c, ε}	{$}

↓
Right end marker

Predictive Parse table :

	a	b	c	$
S	S → aA	S → bB	Error	Error
A	A → aA	A → b	Error	Error
B	Error	B → bA	B → c	B → ε

$S \rightarrow aA \mid BaA$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

	First	follow
S	{a, b, ε}	{$}
A	{a, ε}	{$}
B	{b, ε}	{$}

If there is same symbol then it will not be a LL(1) parser

	a	b	$
S	$S \rightarrow aA$	$B \rightarrow bB$	$B \rightarrow \varepsilon$
A	$A \rightarrow aA$	Error	$A \rightarrow \varepsilon$
B	$B \rightarrow bB$	Error	$B \rightarrow \varepsilon$

FT (L-5)

$E \rightarrow TE'$

$E \rightarrow +TE' | \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' | \varepsilon$

$F \rightarrow (E) | id$ — T

NT

input string : id + id * id

	First	Follow ← End marker
E	{(, id}	{), $}
E'	{+, ε}	{), $}
T	{(, id}	{+,), $}
T'	{*, ε}	{+,), $}
F	{(, id}	{*, +,), $}

Predictive Parse table:

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

Stack movement on predictive parser: (LL1)

Stack	Input	Action
$E	id+id*id$	$E \to TE'$
$E'T	id+id*id$	$T \to FT'$
$E'T'F	id+id*id$	$F \to id$
$E'T'id	id+id*id$	pop id
$E'T'	+id*id$	$T' \to \varepsilon$
$E'	+id*id$	$E' \to +TE'$
$E'T+	+id*id$	pop +
$E'T	id*id$	$T \to FT'$
$E'T'F	id*id$	$F \to id$
$E'T'id	id*id$	pop id
$E'T'	*id$	$T' \to *FT'$
$E'T'F*	*id$	pop *

stack	Input	Action
$E'T'F	id$	$F \to id$
$E'T'id	id$	pop id
$E'T'	$	$T' \to \varepsilon$
$E'	$	$E' \to \varepsilon$
$	$	

End — Right end marker

Successfully parsed

$S \rightarrow (S) \mid \varepsilon$

	first	follow
S	{(, ε}	{), $}

Input string : (())

Parse table :

	(	)	$
S	S→(S)	S→ε	S→ε

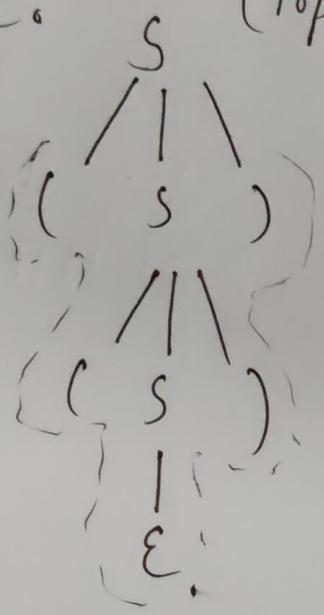
*We will use top down parsing

stack movement :

stack	Input	Action
$S	(())$	S→(S)
$)S(	(())$	pop (
$)S	())$	S→(S)
$))S(	())$	pop (
$))S	))$	S→ε
$))	))$	pop)
$)	)$	pop)
$	$	end

parsed

Parse tree:



(Top down parsing:

* No Left Recursion
* No Ambiguity in grammar
* No common prefix

R.E $(1^*0+01)^*$