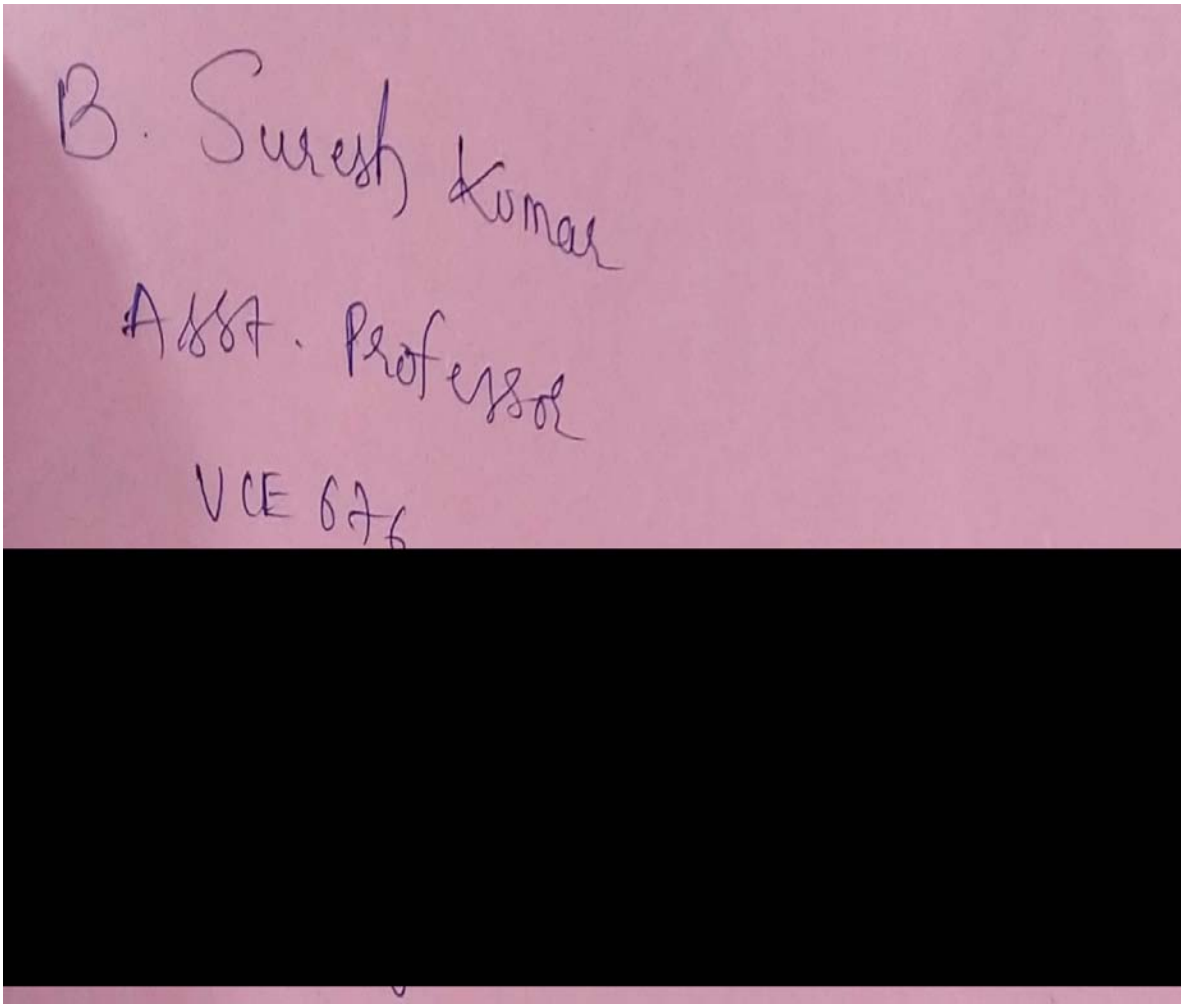
Department of Computer Science & Engineering

Dynamic Activities Conducted by the Faculty Members



→ Tue janMon | Tue jan Tue | Tue

→ Wed jan Tue | Wed

→ Thu feb Wed | Thu

→ mar

mar jan mar jan mar

→ Tue jan Tue [reduce Tue → Wed]

→ Wed jan Tue [Tue → Wed]

→ Thu jan Tue [Wed → Thu]

→ mar jan Tue [Thu → mar]

→ mar jan Tue jan Tue [reduce Tue → Tue jan Tue]

→ mar jan Wed jan Tue [Tue → Wed]

→ mar jan Thu jan Tue [Wed → Thu]

→ mar jan mar jan Tue [Thu → mar]

→ mar jan mar jan Wed [reduce Tue → Wed]

→ mar jan mar jan Thu [Wed → Thu]

→ mar jan mar jan mar [Thu → mar]

stack

mar

Thu

mu jan

I/p

mar jan mar jan mar

jan mar jan mar

jan mar jan mar

mar jan mar

Action

shift

reduce [Thu → mar]

shift

T+E|T

V*T|V

id

id*id+id

Ellipse → Term + Ellipse | Term

Term → Venus * Term | Venus

Venus → id

I/P :- id+id+id

E → T+E |T.

E → T+E (E→T)

E → VT+T (V*T). T→V*T

E → V*T+T (T→V)

E → V*V+T (T→V)

E → V*V+V (V→id)

E → V*V+id. (V→id)

E → V*id+id (V→id)

E → id*id+id (V→id)

E → T+E (E→T)

E → T+T (T→V*T)

E → T+V*T (T→V)

E → T+V*V (V→id)

E → T+V*id (V→id)

E → T+id*id (T→V)

E → V+id*id. (V→id)

E → id+id*id

tack	i/p	Action
$	id+id*id$	shift id
id	+id*id$	reduce V→id
V	+id*id$	shift +
V+	id*id$	shift id
V+id	*id$	reduce V→id
$V+V	*id$	shift *
$V+V*	id$	shift id
$V+V*id	$	reduce V→id
$V+V*V	$	reduce E→T
$V+V*T	$	reduce V→T
$V+T*T	$	reduce T→V*T

8

LO

$T*E $ reduce E→

$E $ acc

S → A+B
→ A+aB;
→ A+ac;
→ bA+ac;
→ baA+ac;
→ bbaA+ac;

[S → A+B]
[B → aB;]
[B → c]
[A → bA]
[A → aA]
[A → b]

I/P

bbaA+ac;
baA+ac;
aA+ac;
A+ac;

Action

shift
reduce A → b
shift
shift

bba+ac;$
ba+ac;
a+ac;
+ac;
+ac;
+ac;
ac;
c;
;$
;$
$
$
$

A → b shift
A → b
shift
A → Ac
A → bA
shift
shift
shift
B → c
shift
B → aB;
S → A+B
Accept

$
$b
$bb
$bA
$bAa
$bA
$A
$A+
$A+a
$A+ac
$A+aB
$A+aB;
$A+B
$S
$

tion table

B. Suresh Kumar

UCE 676

Asst. Prof, CSE

Activity on grammar identification (operator precedence)

Subject: Compiler Design. V Sem

function table.

ABd
a|b
Lc id | id } Find precedence function table ?

TA
+TA/ϵ
FB
*FB/ϵ } Find First and Follow and precedence function table ?
(E)/id

— All THE BEST —

ABd
a|b
Lcid|id

eliminate "L" production rules because it is not useful. So, the production su

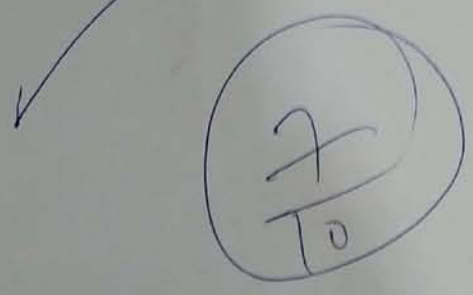
follows

ABd
a|b.

ace A by a|b and substitute in production "S" in order to get the production
ammar format.

→ aBd | bBd
→ a|b.

production rules, so we cannot derive any function table.



2. $E \to TA$
$A \to +TA \mid \epsilon$
$T \to FB$
$B \to *FB \mid \epsilon$
$F \to (E) \mid id$.

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

First(A) =

	First	Follow
$E \to TA$	{(, id}	{$,)}
$A \to +TA \mid \epsilon$	{+, ε}	{$,), +}
$T \to FB$	{(, id}	{$,), +}
$B \to *FB \mid \epsilon$	{*, ε}	{$,), +}
$F \to (E) \mid id$	{(, id}	{$,), *, +}

~~$E \to TA$~~
~~$E \to T+TA \mid T$~~
~~$E \to T+E \mid T$~~

$E \to TA$
$E \to T+TA \mid T$
$E \to T+E \mid T$
$A \to +E \mid \epsilon$
$F \to FB$
$T \to idB \mid (E)B$
$B \to *FB \mid \epsilon$
$B \to *T \mid \epsilon$

$\Rightarrow$

$E \to T+E \mid T$
$A \to +E \mid \epsilon$
$T \to idB \mid (E)B$
$B \to *T \mid \epsilon$
$F \to (E) \mid id$.

Precedence func

	id	+	*	(	)	$
id	–	>	>	<	>	>
+	<	>	<	<	>	>
*	<	>	>	<	>	>
(	<	>	<	<	=	
)	>	>	>		>	>
$	<	<	<	<		

precedence funtion table.

BS|ε

3A|ab

BAb|ε|c

arsing action table (bintb charb)

intL|SL|ε

chars|a|b

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

$A \to \beta A'$

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

$S \to \varepsilon S'$

$S' \to int\ L.S' \mid \varepsilon$

$S \to \varepsilon . S''$

$S'' \to L.S'' \mid \varepsilon$

$L \to aL' \mid bL'$

$L' \to chars\ SL'$

$\Rightarrow$

$S \to S'$

$S' \to int\ LS' \mid \varepsilon$

$S \to S''$

$S'' \to LS'' \mid \varepsilon$

$L \to aL' \mid bL'$

$L' \to chars\ SL'$

	First	Follow
S	int, a, b, ε	$, a, b, int, chars
S'	int, ε	$, a, b, int, chars
S''	a, b, ε	$, a, b, int, chars
L	a, b	$, a, b, int, chars
L'	chars	$, a, b, int, chars

$\Rightarrow$

	a	b	int	chars
S	S→S' / S→S''	S→S' / S→S''	S→S' / S→S''	
S'			S'→intLS'	
S''	S→ε	S→ε	S→ε / S→ε	S→ε
L				
L'				

LL(1) parser can't be designed

& given string can't be validated.

max → max or yum | yum

yum → yeeha and yamama | yamama

yamama → not yamama | (max) | True | false.

Construct a parsing table

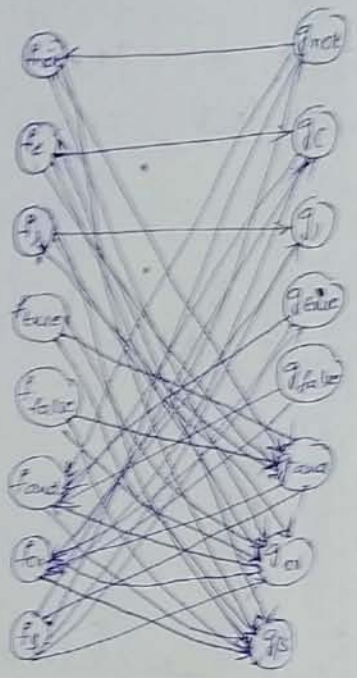
$e \rightarrow te'$

$e' \rightarrow +te' \mid \epsilon$

$t \rightarrow ft'$

$t' \rightarrow *ft' \mid \epsilon$

$f \rightarrow (e) \mid x \mid y$



Solution

max → max or yum | yum

max → yum max'

max' → or yum max' | ε

yum → yeeha and yamama | yamama

yamama → not yamama | (max) | True | False

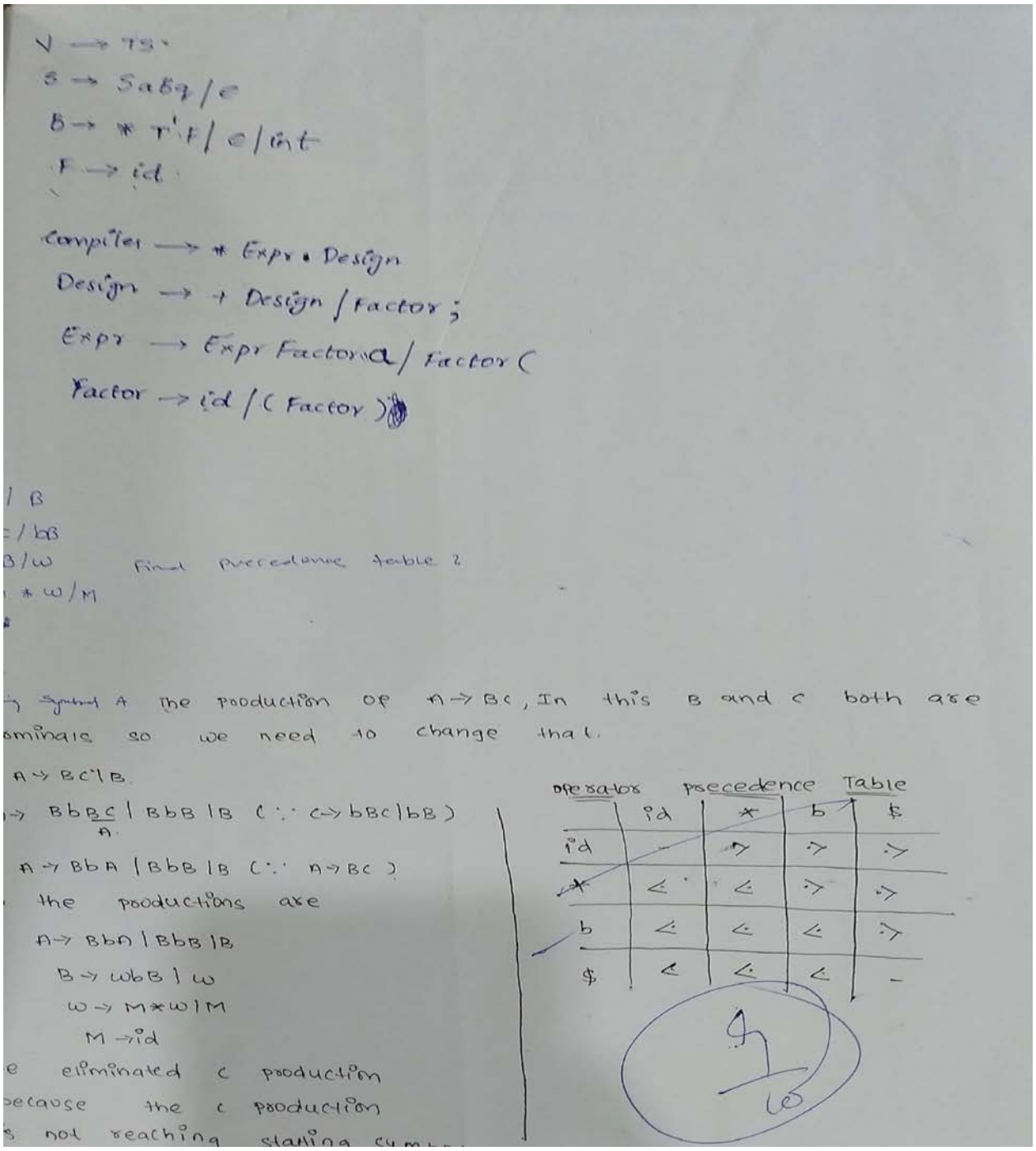
	not	(	)	true	false	and	or	$
not	<	=	=	=	=	>	>	>
(	=	<	=	=	=	>	>	>
)	=	=	>	=	=	>	>	>
true	=	=	=	-	-	>	>	>
false	=	=	=	-	-	>	>	>
and	<	<	<	<	<	>	>	>
or	<	<	<	<	<	<	>	>
$	<	<	<	<	<	<	<	-

	not	(	)	true	false	and	or	$
f	3	0	0	0	0	2	1	0
g	4	0	0	0	0	2	1	0

2

	First	Follow
$e \to te'$	$\{(, x, y\}$	$\{\$,)\}$
$e' \to +te' \mid \epsilon$	$\{+, \epsilon\}$	$\{\$,)\}$
$t \to ft'$	$\{(, x, y\}$	$\{\$,), +\}$
$t' \to *ft' \mid \epsilon$	$\{*, \epsilon\}$	$\{\$,), +\}$
$f \to (e) \mid x \mid y$	$\{(, x, y\}$	$\{\$,), +, *\}$

	x	y	(	)	+	*	$
e	$e \to te'$	$e \to te'$	$e \to te'$	error	error	error	error
e'				$e' \to \epsilon$	$e' \to +te'$	error	$e' \to \epsilon$
t	$t \to ft'$	$t \to ft'$	$t \to ft'$				
t'				$t' \to \epsilon$	$t' \to \epsilon$	$t' \to *ft'$	$t' \to \epsilon$
f	$f \to x$	$f \to y$	$f \to (e)$				



Faculty Name: Mr. B. Suresh Kumar
Activity Name: TAPPS
Class: B.Tech III Year CSE
Subject: Compiler Design
Topic: LR grammars, LR parsers-simple LR