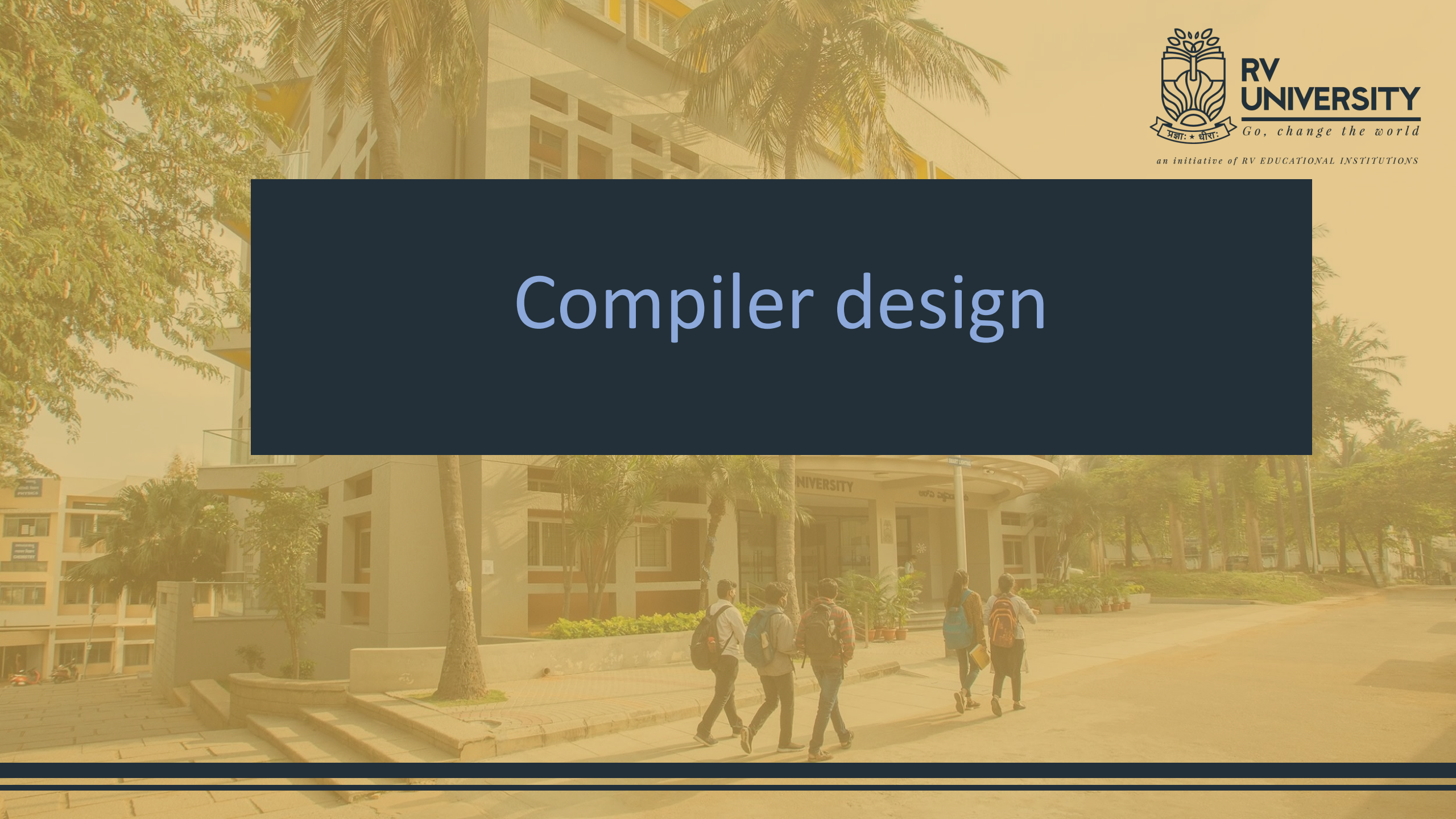
RV UNIVERSITY

Go, change the world

an initiative of RV EDUCATIONAL INSTITUTIONS

NEW-AGE GLOBAL UNIVERSITY FOR LIBERAL EDUCATION



Compiler design

SYNTAX DIRECTED TRANSLATION

SYNTAX DIRECTED TRANSLATION ASSOCIATES INFORMATION WITH A GRAMMAR BY ATTACHING ATTRIBUTES TO THE GRAMMAR SYMBOLS AND VALUES FOR ATTRIBUTE ARE COMPUTED BY SEMANTIC RULES ASSOCIATED WITH THE GRAMMAR PRODUCTION.

THERE ARE TWO WAYS FOR ASSOCIATING INFORMATION TO THE PRODUCTIONS OF A GRAMMAR

1) SYNTAX DIRECTED DEFINITION (SDD)
2) SYNTAX DIRECTED TRANSLATION SCHEME (SDT)

Syntax Directed Definition

IT IS AN AUGMENTED CONTEXT FREE GRAMMAR IN WHICH EACH GRAMMAR SYMBOL HAS AN ASSOCIATED SET OF ATTRIBUTES CALLED SYNTHESIZED ATTRIBUTES & INHERITED ATTRIBUTES AND SEMANTIC RULES FOR COMPUTING THE VALUES OF THE ATTRIBUTES ASSOCIATED WITH THE SYMBOLS APPEARING IN THE PRODUCTIONS.

A PARSE TREE SHOWING THE ATTRIBUTE VALUES AT EACH NODE IS CALLED AN ANNOTATED PARSE TREE.

SDD for a Simple Calculator

Productions & Semantic Rules

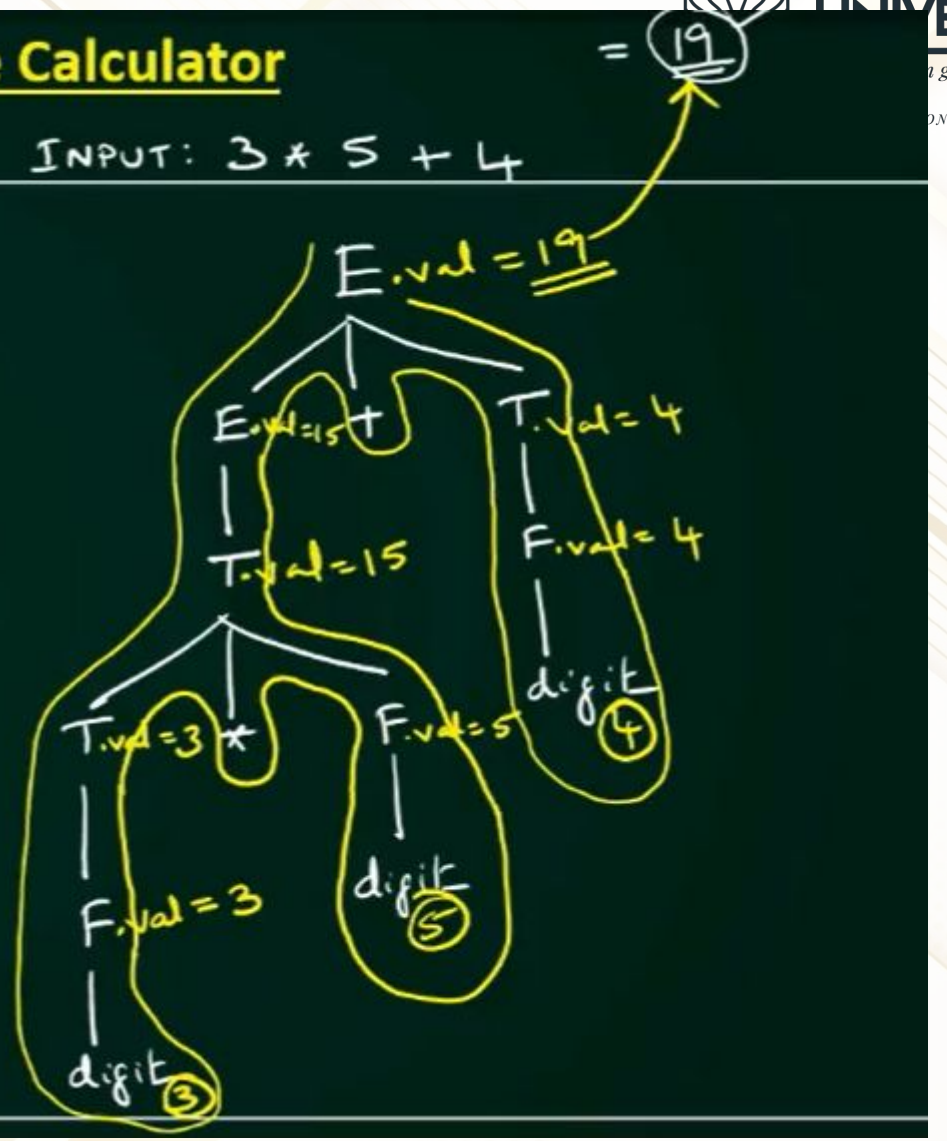
$E \rightarrow E + T$	$E.val = E.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T * F$	$T.val = T.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow digit$	$F.val = digit$

SDD for a Simple Calculator

Productions & Semantic Rules

Production	Semantic Rule
$E \rightarrow E + T$	$E.val = E.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T * F$	$T.val = T.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow digit$	$F.val = digit$

INPUT: 3 * 5 + 4



SDD

Productions & Semantic Rules

$S \rightarrow x\,x\,W$	Printf("1")
$\mid y$	Printf("2")
$W \rightarrow S_3$	Printf("3")

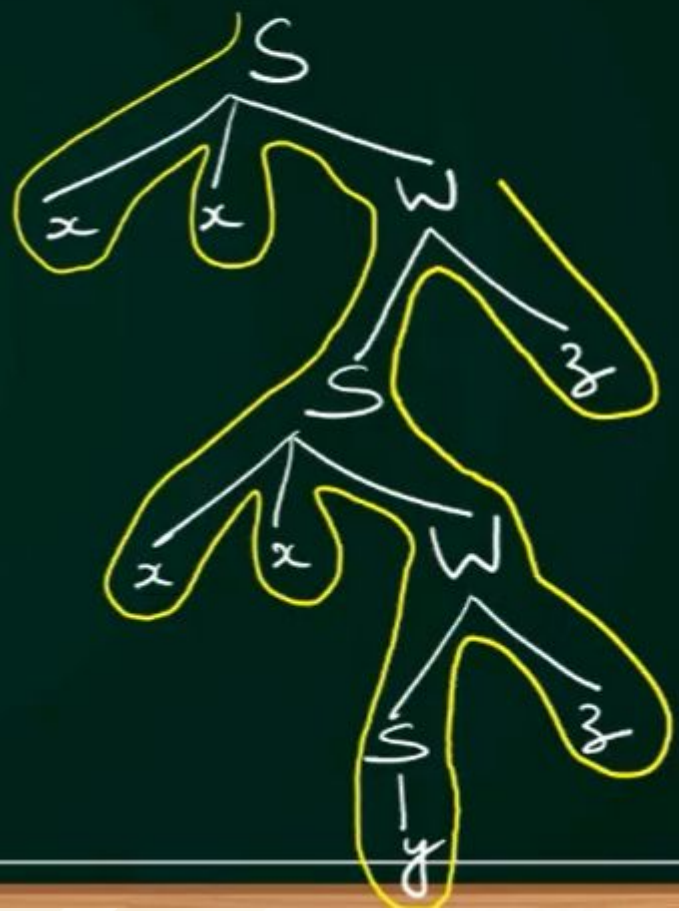
SDD

O/P = 231

INPUT : x x x x y 3 3

Productions & Semantic Rules

$S \rightarrow x\,x\,W$	Printf("1")
$\mid y$	Printf("2")
$W \rightarrow S\,3$	Printf("3")



SDD

Productions & Semantic Rules

INPUT: 2 # 3 & 5 # 6 & 4

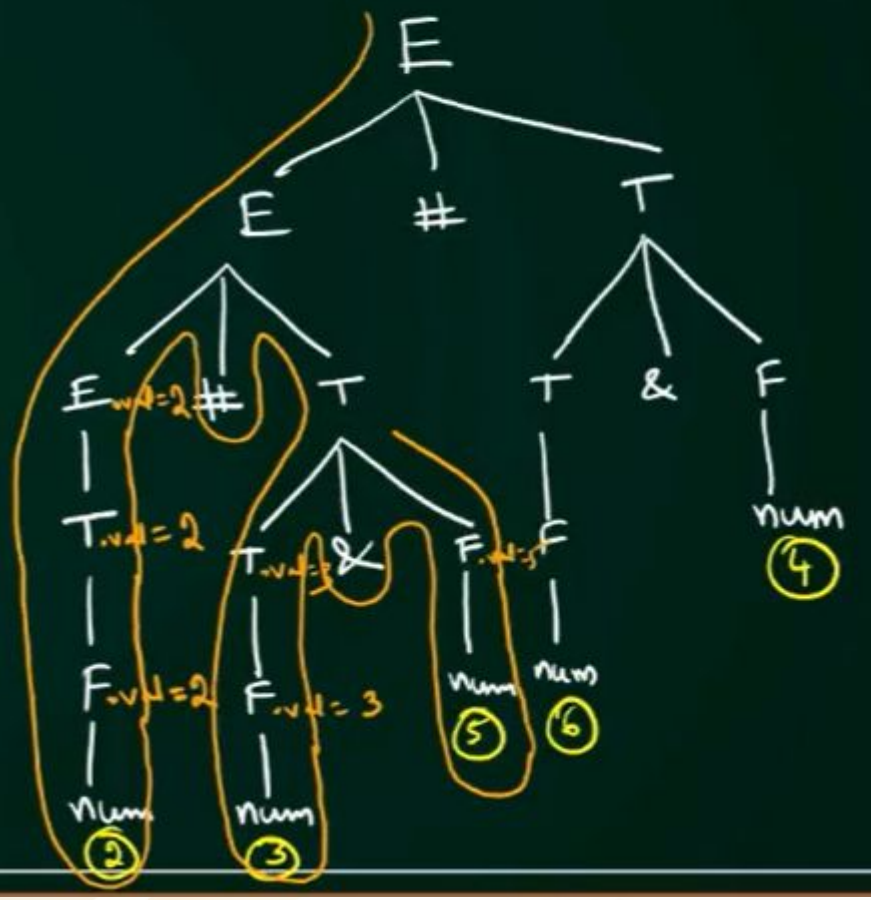
$E \rightarrow E \# T$	$E.val = E.val * T.val$
$\mid T$	$E.val = T.val$
$T \rightarrow T \& F$	$T.val = T.val + F.val$
$\mid F$	$T.val = F.val$
$F \rightarrow num$	$F.val = num.lvalue$

SDD

Productions & Semantic Rules

$E \rightarrow E \# T$	$E.val = E.val * T.val$
$\mid T$	$E.val = T.val$
$T \rightarrow T \& F$	$T.val = T.val + F.val$
$\mid F$	$T.val = F.val$
$F \rightarrow num$	$F.val = num.lvalue$

INPUT: $(2 \overset{*}{\#} (3 \overset{+}{\&} 5) \overset{*}{\#} (6 \overset{+}{\&} 4)) = 160$



SDD

Productions & Semantic Rules

INPUT: $((2+3)==8)$

$E \rightarrow E_1 + E_2$	if $((E_1.type == E_2.type) \&\& (E_1.type == int))$ then $E.type = int$ else ERROR
$\mid E_1 == E_2$	if $((E_1.type == E_2.type) \&\& (E_1.type == int/bool))$ then $E.type = bool$ else ERROR
$\mid (E_1)$	$E.type = E_1.type$
$\mid num$	$E.type = int$
$\mid True$	$E.type = bool$
$\mid False$	$E.type = bool$

Productions & Semantic Rules

INPUT = 1011

$N \rightarrow L$	$N.count = L.count$
$L \rightarrow L_1 B$	$L.count = L_1.count + B.count$
$\mid B$	$L.count = B.count$
$B \rightarrow 0$	$B.count = 0$
$\mid 1$	$B.count = 1$

SDD

Productions & Semantic Rules

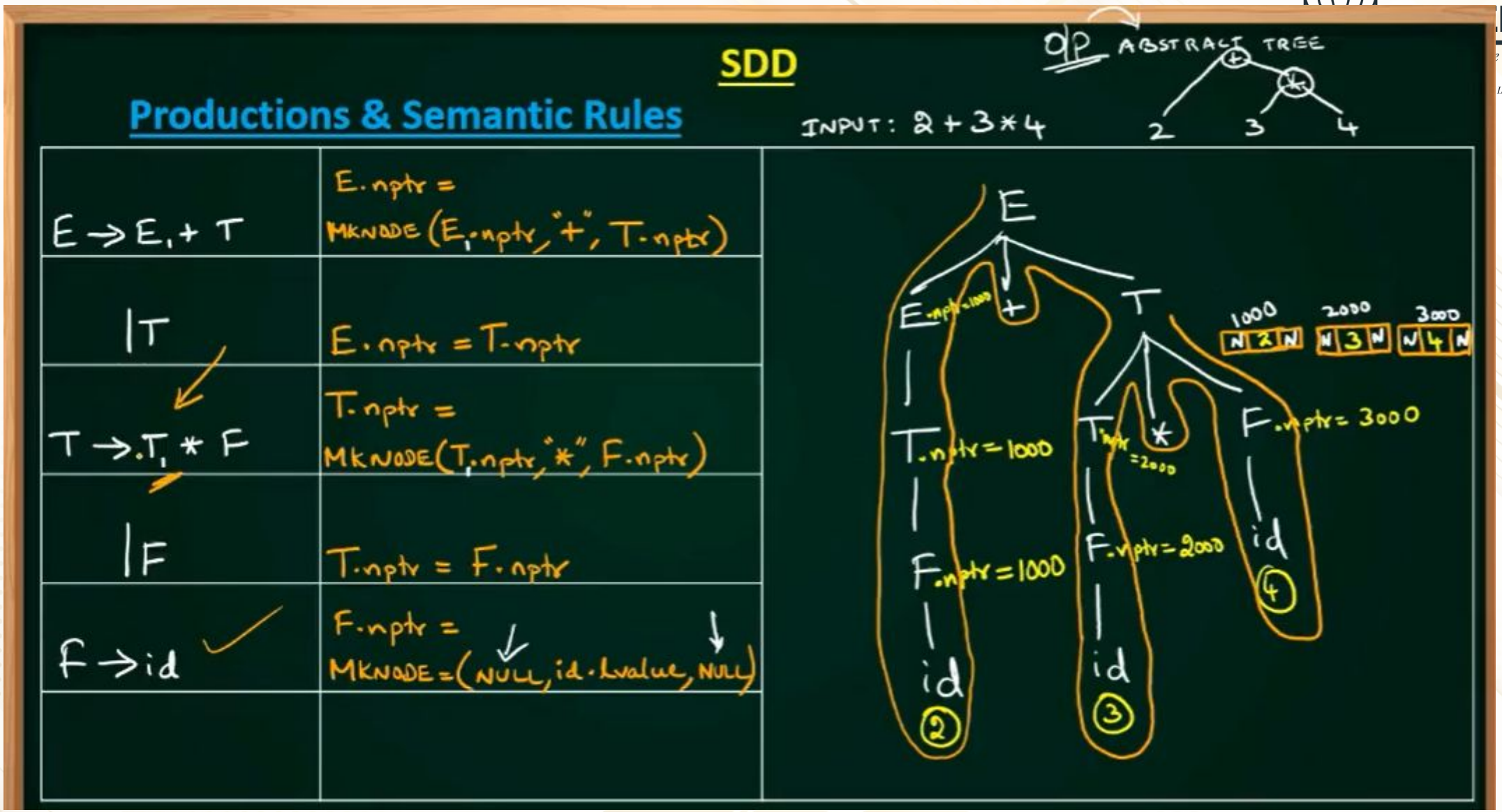
$N \rightarrow L$	$N.count = L.count$
$L \rightarrow L_1 B$	$L.count = L_1.count + B.count$
$\mid B$	$L.count = B.count$
$B \rightarrow 0$	$B.count = 0$
$\mid 1$	$B.count = 1$

INPUT = 1011

SDD

Productions & Semantic Rules

$S \rightarrow id = E$	GEN(id.name = E.place)
$E \rightarrow E_1 + T$	E.place = NEW Temp() GEN(E.place = "E_1.place + T.place)
$\mid T$	E.place = T.place
$T \rightarrow T * F$	T.place = NEW Temp() GEN(T.place = "T_1.place * F.place)
$\mid F$	T.place = F.place
$F \rightarrow id$	F.place = id.name

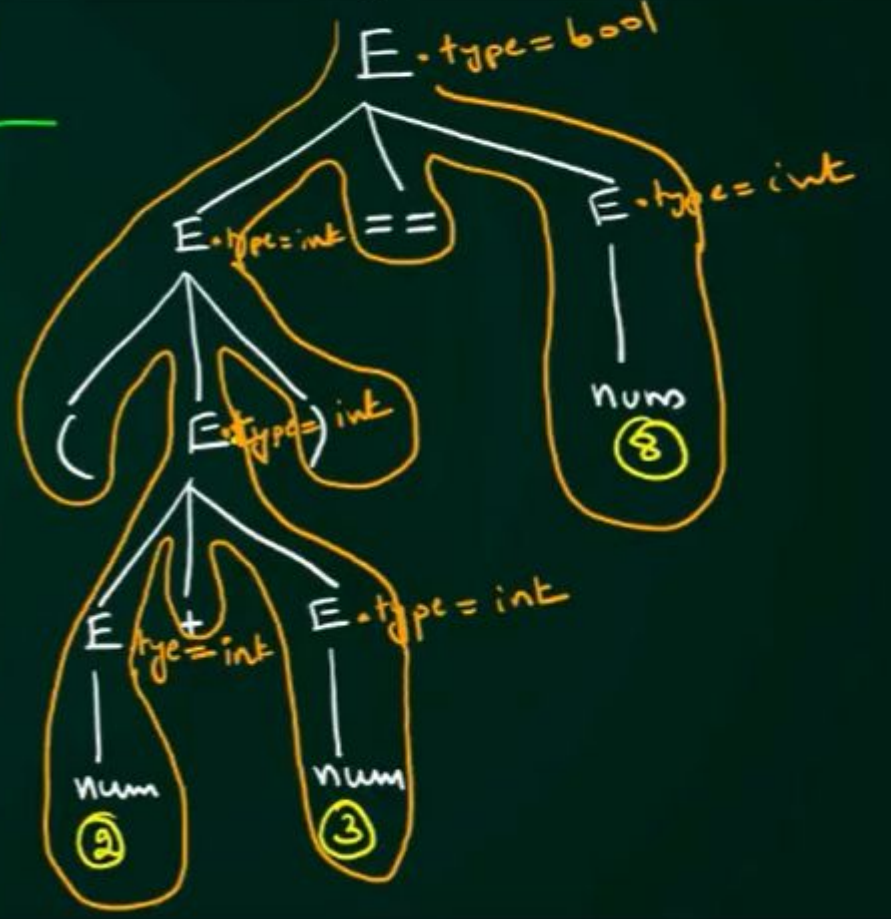


SDD

Productions & Semantic Rules

$E \rightarrow E_1 + E_2$	if $((E_1.type == E_2.type) \&\& (E_1.type == int))$ then E.type = int else ERROR
$\mid E_1 == E_2$	if $((E_1.type == E_2.type) \&\& (E_1.type == int/bool))$ then E.type = bool else ERROR
$\mid (E_1)$	E.type = E_1.type
| num	E.type = int
| True	E.type = bool
| False	E.type = bool

INPUT: $(2 + 3) == 8$

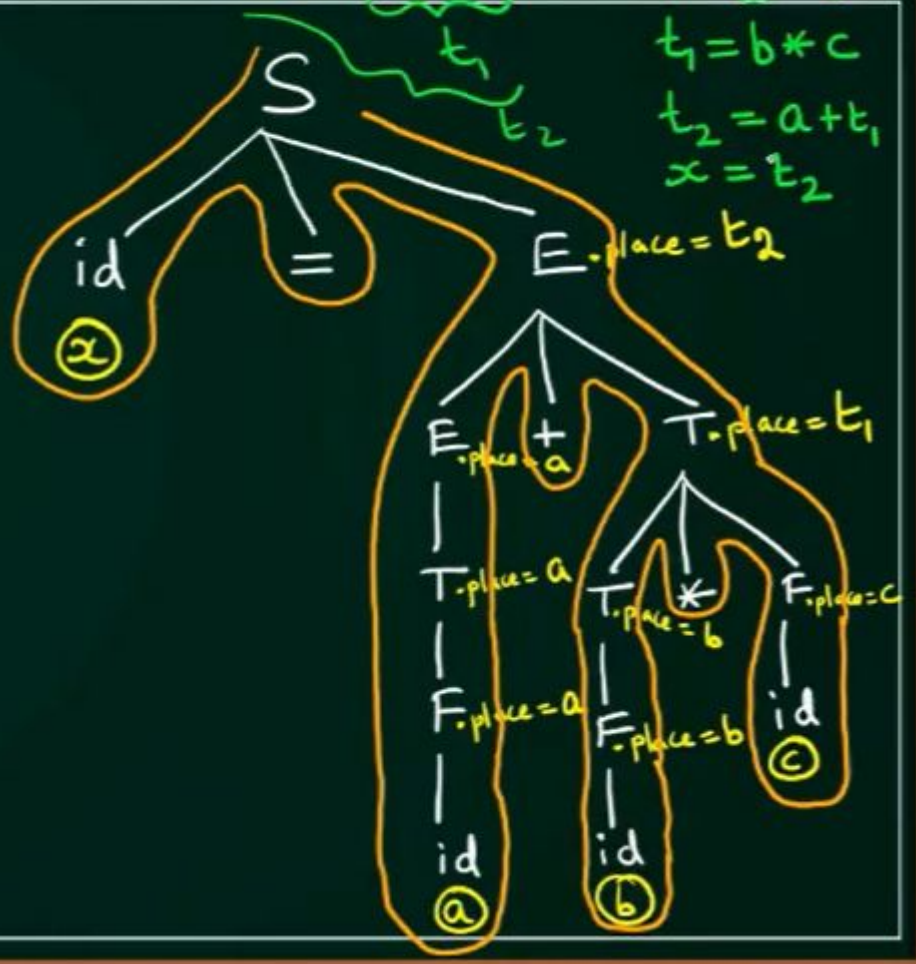


SDD FOR GENERATING THREE ADDRESS CODE

Productions & Semantic Rules

Production	Semantic Rule
** $S \rightarrow id = E$	GEN(id.name "=" E.place)
$E \rightarrow E_1 + T$	E.place = NEW Temp( ) GEN(E.place "=" E_1.place "+" T.place)
$\mid T$	E.place = T.place
$T \rightarrow T_1 * F$	T.place = NEW Temp( ) GEN(T.place "=" T_1.place "*" F.place)
$\mid F$	T.place = F.place
$F \rightarrow id$	F.place = id.name

INPUT: $x = a + b * c$



Productions & Semantic Rules

INPUT : 9 − 5 + 2 OUTPUT = 9 5 − 2 +

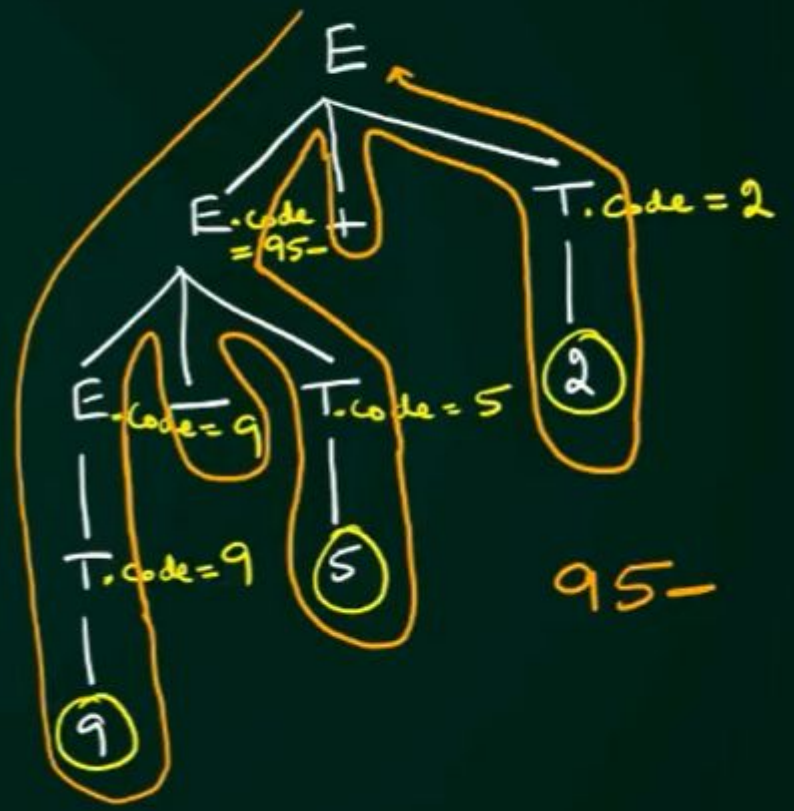
$E \rightarrow E_1 + T$	$E.code = E_1.code \| T.code \| '+'$
$E \rightarrow E_1 - T$	$E.code = E_1.code \| T.code \| '-'$
$E \rightarrow T$	$E.code = T.code$
$T \rightarrow 0$	$T.code = '0'$
$T \rightarrow 1$	$T.code = '1'$
⋮	⋮
$T \rightarrow 9$	$T.code = '9'$

SDD FOR CONVERTING INFIX TO POST FIX

INPUT : 9 − 5 + 2 OUTPUT = 95−2+

Productions & Semantic Rules

Productions	Semantic Rules
$E \rightarrow E_1 + T$	$E.code = E_1.code \| T.code \| '+'$
$E \rightarrow E_1 - T$	$E.code = E_1.code \| T.code \| '-'$
$E \rightarrow T$	$E.code = T.code$
$T \rightarrow 0$	$T.code = '0'$
$T \rightarrow 1$	$T.code = '1'$
⋮	⋮
$T \rightarrow 9$	$T.code = '9'$



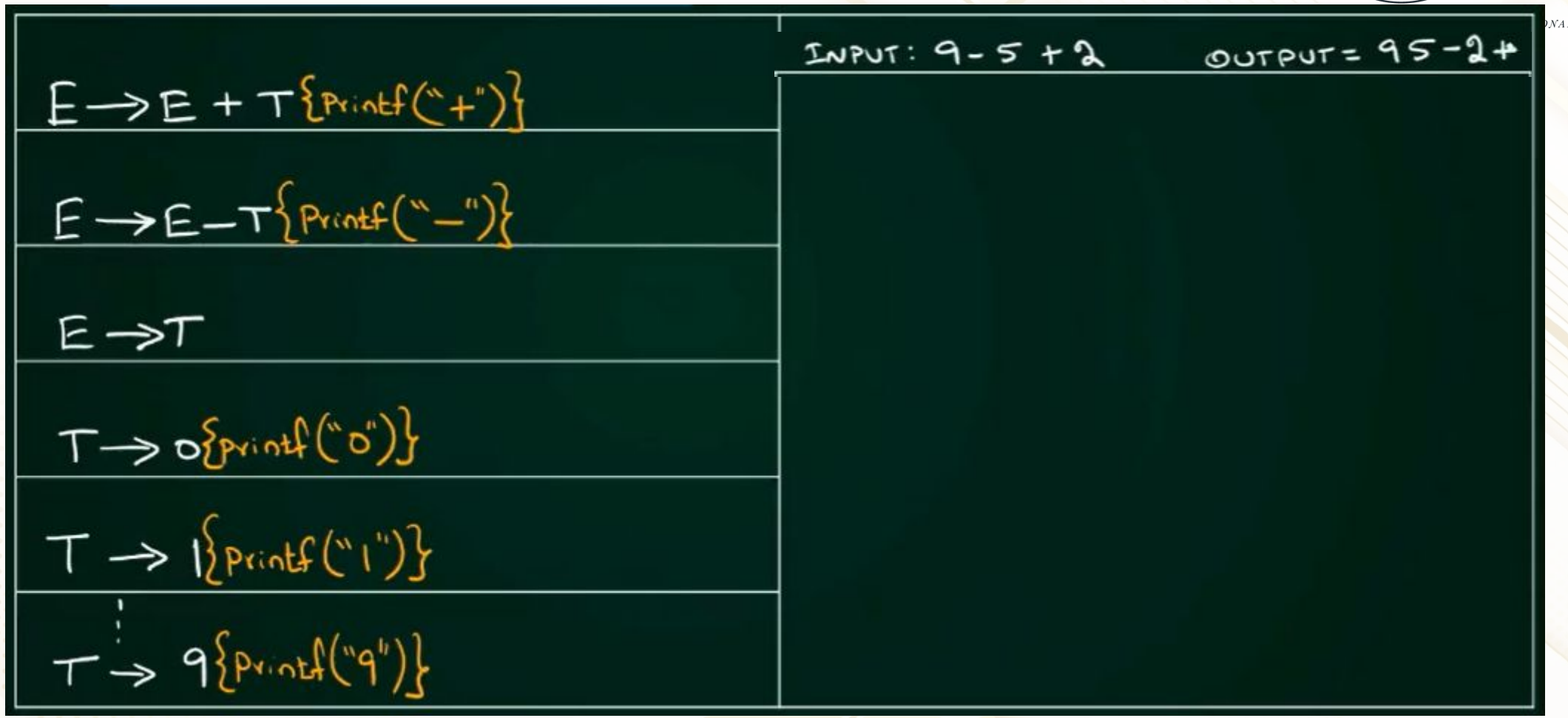
Syntax Directed Translation Scheme

IT IS A CONTEXT FREE GRAMMAR IN WHICH SEMANTIC ACTIONS(PROGRAM FRAGMENTS) ARE ENCLOSED BETWEEN BRACES OR INSERTED WITHIN THE RHS OF A PRODUCTION.

TRANSLATION SCHEME IS LIKE A SYNTAX DIRECTED DEFINITION EXCEPT THAT THE ORDER OF EVALUATION OF SEMANTIC RULES IS EXPLICITLY SHOWN.

A TRANSLATION SCHEME GENERATES AN OUTPUT FOR EACH SENTENCE OF THE GRAMMAR BY EXECUTING THE ACTIONS IN THE ORDER THEY APPEAR DURING A DEPTH FIRST TRAVERSAL(FROM LEFT TO RIGHT) OF A PARSE TREE.

Productions and Semantic Action



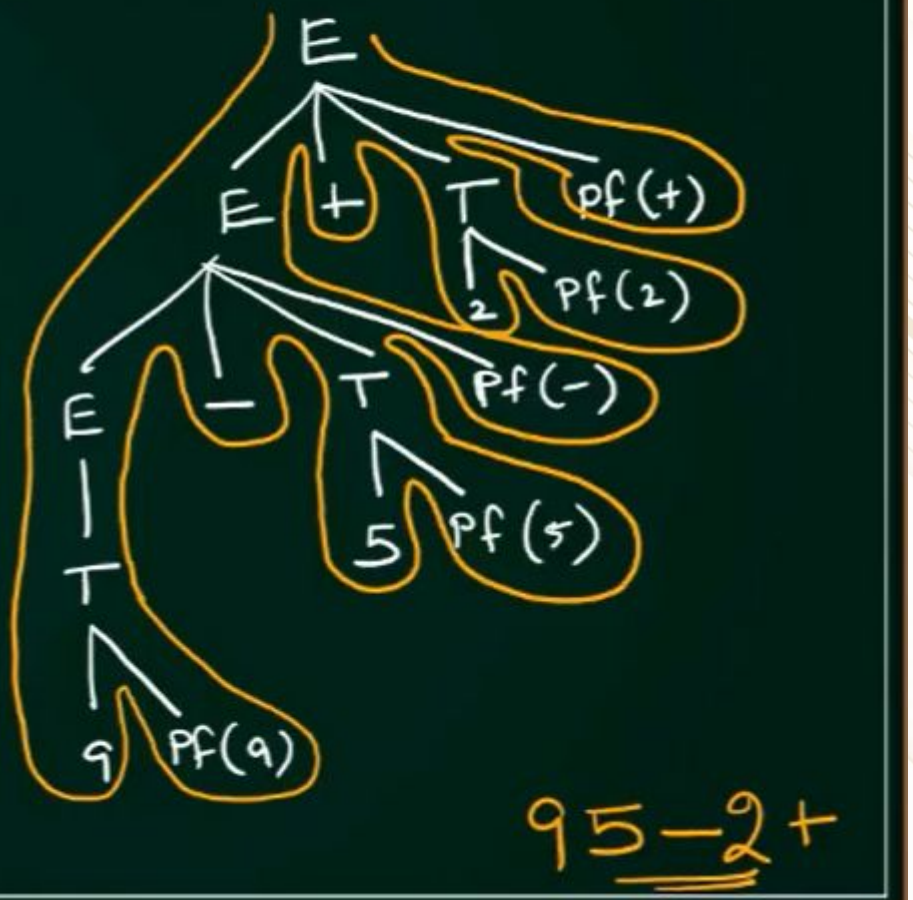
SEMANTIC ACTIONS

SDT FOR CONVERTING INFIX TO POST FIX

Productions & Semantic Rules

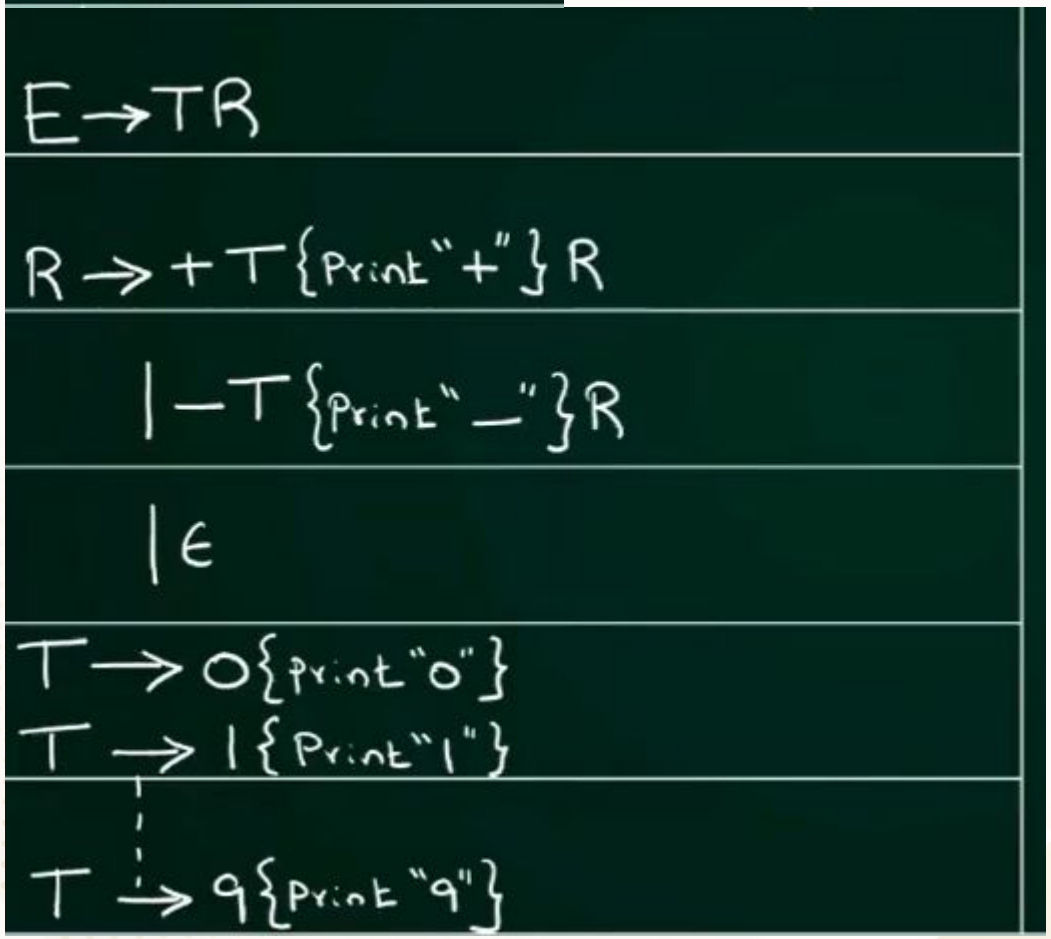
INPUT: 9-5+2 OUTPUT = 95-2+

$E \rightarrow E + T \{Printf("+")\}$
$E \rightarrow E - T \{Printf("-")\}$
$E \rightarrow T$
$T \rightarrow 0 \{Printf("0")\}$
$T \rightarrow 1 \{Printf("1")\}$
$\vdots$
$T \rightarrow 9 \{Printf("9")\}$



SDT

INPUT: 3+5−4

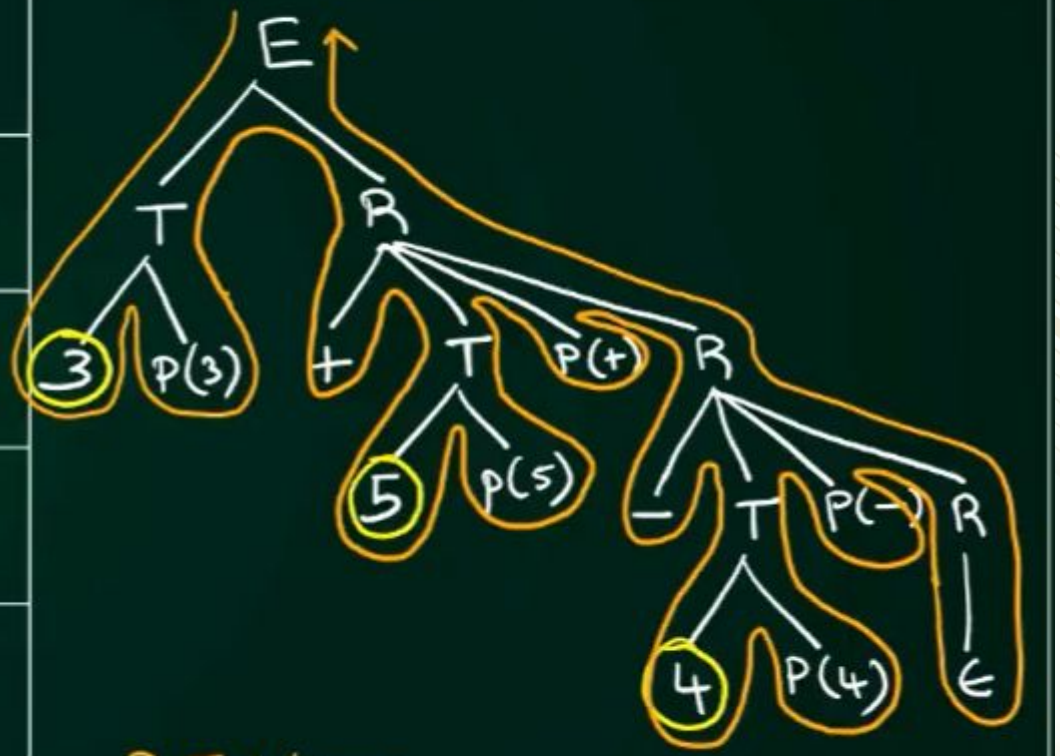


SDT

Productions & Semantic Rules (Actions)

Productions & Semantic Rules (Actions)
$E \rightarrow TR$
$R \rightarrow +T\{Print"+"\}R$
$\mid -T\{Print"-"\}R$
$\mid \epsilon$
$T \rightarrow 0\{Print"0"\}$ $T \rightarrow 1\{Print"1"\}$
$T \rightarrow 9\{Print"9"\}$

INPUT: 3+5−4



35+4−

SYNTHESIZED ATTRIBUTES

AN ATTRIBUTE IS SAID TO BE SYNTHESIZED IF ITS VALUE AT A PARSE TREE NODE IS DETERMINED FROM ATTRIBUTE VALUES OF THE CHILDREN OF THAT NODE.

SYNTHESIZED ATTRIBUTES HAVE THE PROPERTY THAT THEY CAN BE EVALUATED DURING A BOTTOM UP TRAVERSAL OF THE PARSE TREE.

A SYNTAX DIRECTED DEFINITION THAT USES SYNTHESIZED ATTRIBUTES EXCLUSIVELY IS SAID TO BE AN S-ATTRIBUTED GRAMMAR/DEFINITION.

INHERITED ATTRIBUTES

AN ATTRIBUTE IS SAID TO BE INHERITED IF ITS VALUE AT A PARSE TREE NODE IS DETERMINED FROM THE ATTRIBUTE VALUES AT A PARENT AND/OR SIBLINGS OF THAT NODE.

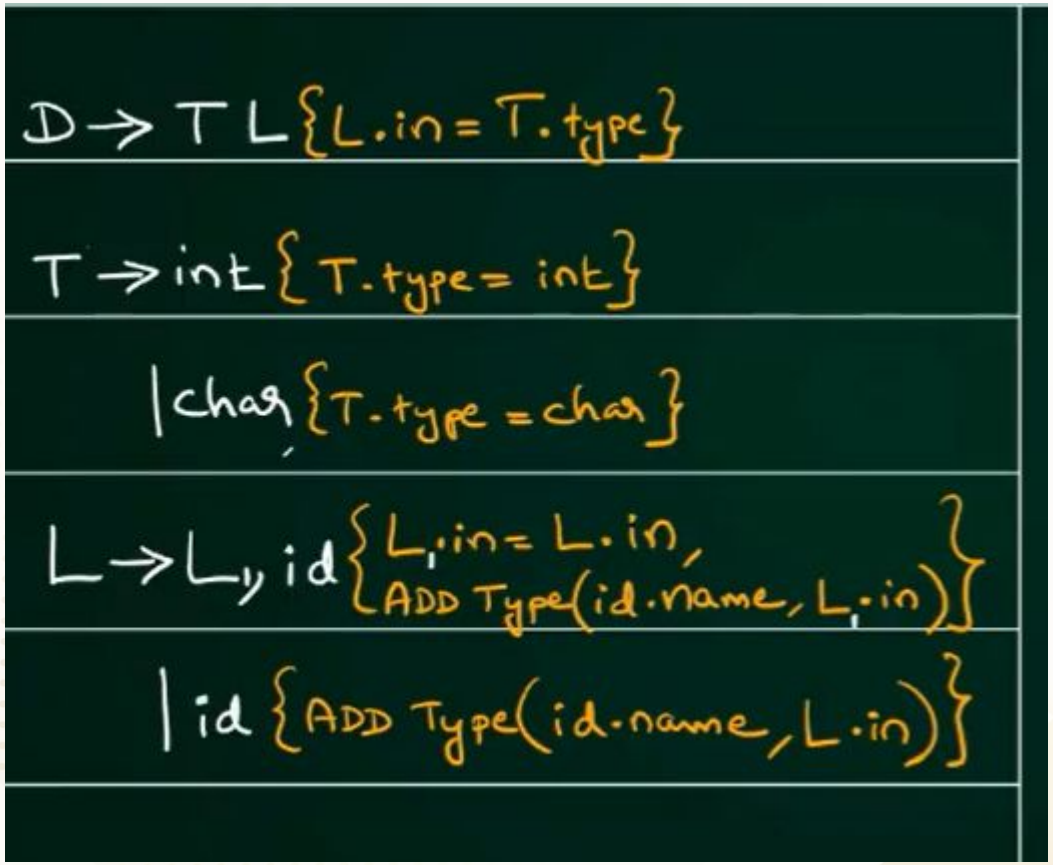
A SYNTAX DIRECTED DEFINITION THAT USES SYNTHESIZED ATTRIBUTES AND/OR INHERITED ATTRIBUTES (RESTRICTED TO INHERIT EITHER FROM PARENT OR LEFT SIBLING ONLY) IS CALLED L-ATTRIBUTED GRAMMAR/DEFINITION.

Difference Between S-Attributed and L-Attributed SDT

S-Attributed SDT	L-Attributed SDT
USES ONLY SYNTHESIZED ATTRIBUTES	USES BOTH SYNTHESIZED & INHERITED ATTRIBUTES EACH INHERITED ATTRIBUTE IS RESTRICTED TO INHERIT EITHER FROM PARENT OR LEFT SIBLING ONLY
SEMANTIC ACTIONS ARE PLACED AT RIGHT END OF PRODUCTION	SEMANTIC ACTIONS ARE PLACED ANYWHERE IN RHS
ATTRIBUTES ARE EVALUATED DURING BOTTOM UP PARSING	ATTRIBUTES ARE EVALUATED BY TRAVERSING PARSE TREE, DEPTH FIRST LEFT TO RIGHT

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

EVALUATE THE SYNTHESIZED ATTRIBUTE WHEN YOU LAST VISIT
EVALUATE THE INHERITED ATTRIBUTE WHEN YOU FIRST VISIT



RV UNIVERSITY

Go, change the world

an initiative of RV EDUCATIONAL INSTITUTIONS

INTERMEDIATE CODE GENERATION

REPRESENTING A SOURCE PROGRAM IN THE INTERMEDIATE FORM HAS THE FOLLOWING TWO ADVANTAGE

1) IT CAN BE TARGETED TO ANY MACHINE
2) MACHINE INDEPENDENT OPTIMIZATION CAN BE PERFORMED

Forms of Intermediate Code

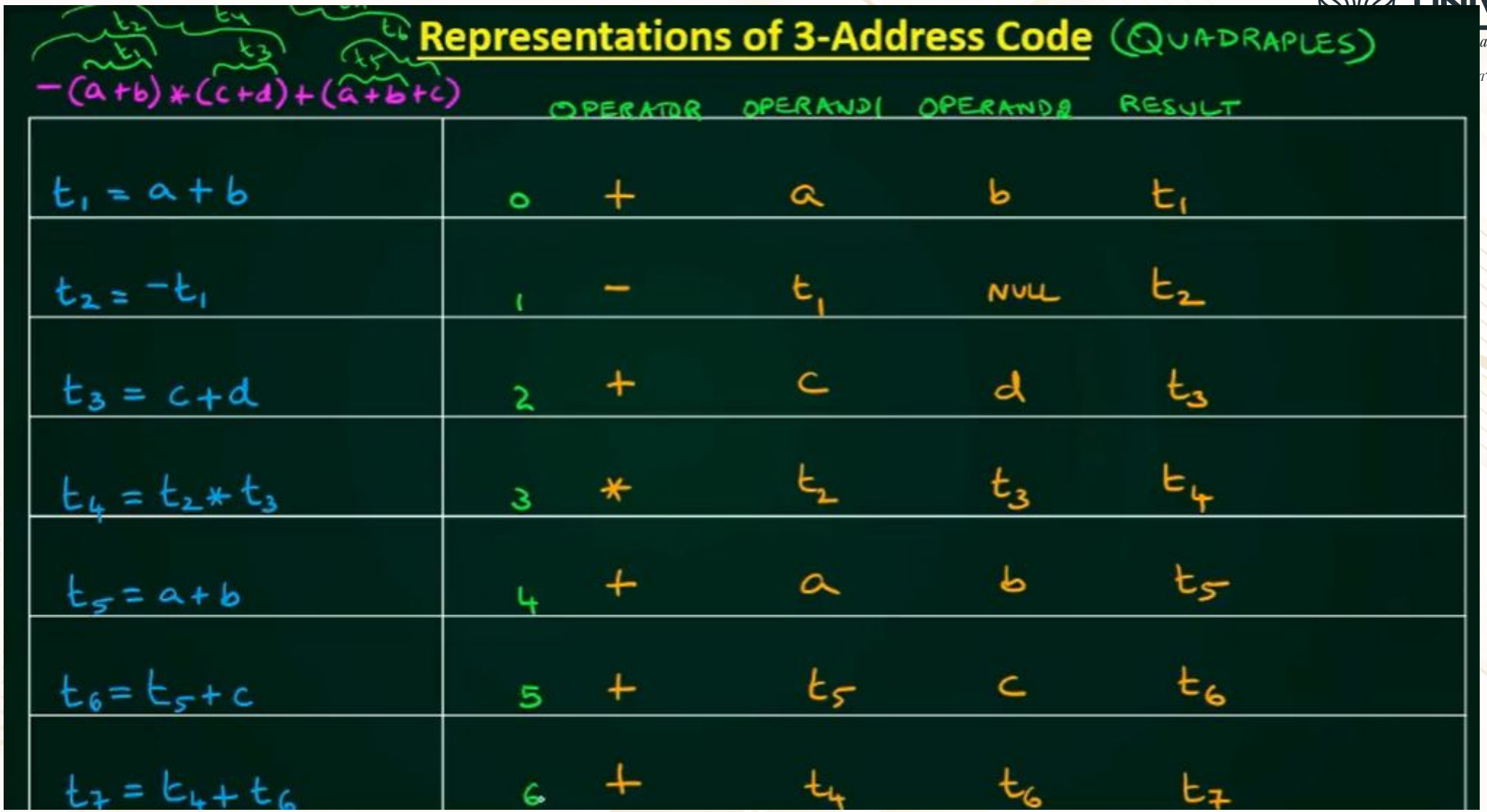
Types of 3-Address Code

$x = y \text{ op } z$
$x = \text{op } z$
$x = y$
if x (rel_op) y goto L
goto L
A[i] = x ; x = A[i]
x = &y ; z = *x ; P = q

Representations of 3-Address Code (QUADRAPLES)

$-(a+b)*(c+d)+(a+b+c)$

$t_1 = a + b$	
$t_2 = -t_1$	
$t_3 = c + d$	
$t_4 = t_2 * t_3$	
$t_5 = a + b$	
$t_6 = t_5 + c$	
$t_7 = t_4 + t_6$	


Representations of 3-Address Code (QUADRAPLES)
$-(a+b)*(c+d)+(a+b+c)$
t_1 t_2 t_3 t_4 t_5 t_6 t_7
OPERATOR OPERAND1 OPERAND2 RESULT
$t_1 = a + b$ 0 + a b t_1
$t_2 = -t_1$ 1 − t_1 NULL t_2
$t_3 = c + d$ 2 + c d t_3
$t_4 = t_2 * t_3$ 3 * t_2 t_3 t_4
$t_5 = a + b$ 4 + a b t_5
$t_6 = t_5 + c$ 5 + t_5 c t_6
$t_7 = t_4 + t_6$ 6 + t_4 t_6 t_7

Representations of 3-Address Code (INDIRECT TRIPLE)

$-(a+b)*(c+d)+(a+b+c)$

	OPERATOR	OPERAND1	OPERAND2	INSTRUCTION POINTERS	
$t_1 = a+b$	0	+	a	b	35	0
$t_2 = -t_1$	1	−	(0)	NULL	36	1
$t_3 = c+d$	2	+	c	d	37	2
$t_4 = t_2 * t_3$	3	*	(1)	(2)	38	3
$t_5 = a+b$	4	+	a	b	39	4
$t_6 = t_5 + c$	5	+	(4)	c	40	5
$t_7 = t_4 + t_6$	6	+	(3)	(5)	50	6

[1] Backpatching and Conversion to 3-Address Code

```
if (a<b) then t = 1
else t = 0
```

3 ADDRESS CODE

```
(i): if a<b goto ____
(i+1): t=0
(i+2): goto ____
(i+3): t=1
(i+4): return
```

[2] Backpatching and Conversion to 3-Address Code

a<b AND c<d OR e<f

```
100: if (a<b) goto ____
101: t1 = 0
102: goto ____
103: t1 = 1
104: if (c<d) goto ____
105: t2 = 0
106: goto ____
107: t2 = 1
108: if (e<f) goto ____
109: t3 = 0
110: goto ____
111: t3 = 1
112: t4 = t1 AND t2
113: t5 = t4 OR t3
```

[3] Backpatching and Conversion to 3-Address Code

while E do S

```
L: if(E==0) goto ___
   S
   goto ___
L1:
```

```
L: if(E) goto ___
   goto ___
L1: S
   goto ___
LAST:
```

[4] Backpatching and Conversion to 3-Address Code

```
while (a<b) do
  x = y + z
```

```
L: if a<b goto ___
   goto ___

L1: t = y + z
    x = t
    goto ___

LAST:
```

[5] Backpatching and Conversion to 3-Address Code

```
for(i=0; i<10; i++)
    a = b + c
```

```
    i = 0
L:  if(i<10) goto ___
    goto ___
L1: t1 = b + c
    a = t1
    t = i + 1
    i = t
    goto ___
LAST:
```