CC Lecture 22

Prepared for: 7th Sem, CE, DDU

Prepared by: Niyati J. Buch

Example 2

Binary to Decimal

N→L.L

L→BL | B

B →0|1

- AS(N)= AS(B) ={val↑:real}
- AS(L) ={cnt↑:integer, val↑:real}

1. N → $L_1.L_2$ {N.val = L_1.val + (L_2.val / 2^L_2.cnt)}
2. L → BL_1 {L.cnt=L_1.cnt+1; L.val=L_1.val+(B.val* 2^L_1.cnt)}
3. L → B {L.cnt = 1 ; L.val = B.val}
4. B → 0 {B.val = 0}
5. B → 1 {B.val = 1}

Example 2 (second method)

Binary to Decimal

N→L.L

L→LB | B

B →0|1

- AS(N)={val↑:real}
- AS(L) = AS(B) ={cnt↑:integer, val↑:real}

1. N → $L_1.L_2$ {N.val = L_1.val + (L_2.val / 2^L_2.cnt)}
2. L → L_1B {L.cnt=L_1.cnt+1; L.val=L_1.val*2 + B.val}
3. L → B {L.cnt = 1 ; L.val = B.val}
4. B → 0 {B.val = 0}
5. B → 1 {B.val = 1}

Example 2 (second method)

1. $N \rightarrow L_1.L_2$ {N.val = L_1.val + (L_2.val / 8^L_2.cnt)} Octal to Decimal
2. $L \rightarrow L_1 B$ {L.cnt=L_1.cnt+1; L.val=L_1.val*8 + B.val}
3. $L \rightarrow B$ {L.cnt = 1 ; L.val = B.val}
4. $B \rightarrow 0$ {B.val = 0}
5. $B \rightarrow 1$ {B.val = 1}
6. $B \rightarrow 2$ {B.val = 2}
7. $B \rightarrow 3$ {B.val = 3}
8. $B \rightarrow 4$ {B.val = 4}
9. $B \rightarrow 5$ {B.val = 5}
10. $B \rightarrow 6$ {B.val = 6}
11. $B \rightarrow 7$ {B.val = 7}

Example 2 (second method)

Hexadecimal to Decimal

1. $N \rightarrow L_1.L_2$ {N.val = L_1.val + (L_2.val / 16^L_2.cnt)}
2. $L \rightarrow L_1 B$ {L.cnt=L_1.cnt+1; L.val=L_1.val*16 + B.val}
3. $L \rightarrow B$ {L.cnt = 1 ; L.val = B.val}
4. B → 0 {B.val = 0}
5. B → 1 {B.val = 1}
6. B → 2 {B.val = 2}
7. B → 3 {B.val = 3}
8. B → 4 {B.val = 4}
9. B → 5 {B.val = 5}
10. B → 6 {B.val = 6}
11. B → 7 {B.val = 7}
12. B → 8 {B.val = 8}
13. B → 9 {B.val = 9}
14. B → a {B.val = 10}
15. B → b {B.val = 11}
16. B → c {B.val = 12}
17. B → d {B.val = 13}
18. B → e {B.val = 14}
19. B → f {B.val = 15}

Examples so far

1. $L = \{a^n b^n c^n \mid n \geq 1\}$ for input aabbcc
2. Binary to decimal (octal to decimal, hexadecimal to decimal)
3. Simple calculator
4. Infix to postfix

SDT for desktop calculator

E → E + T

E → T

T → T * F

T → F

F → X ^ F

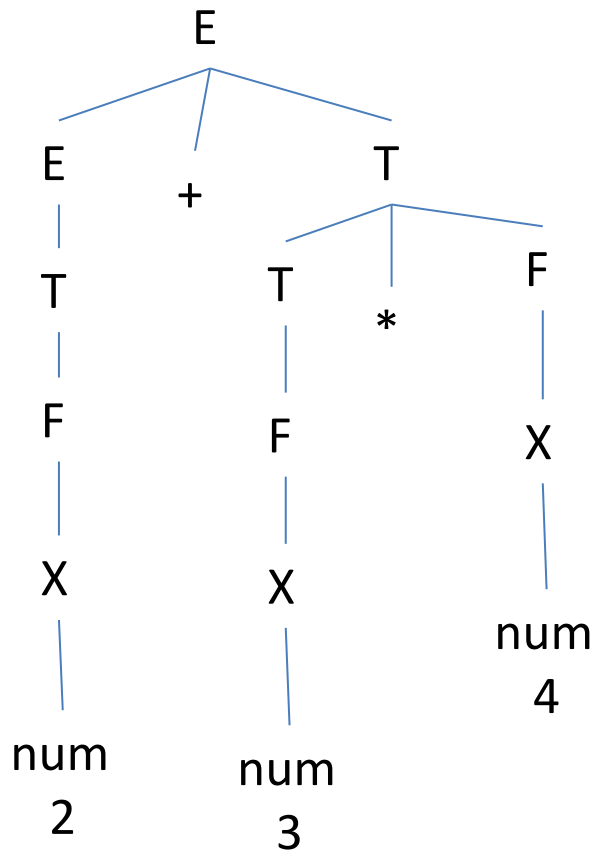
F → X

X → num

SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

For input, 2 + 3 * 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

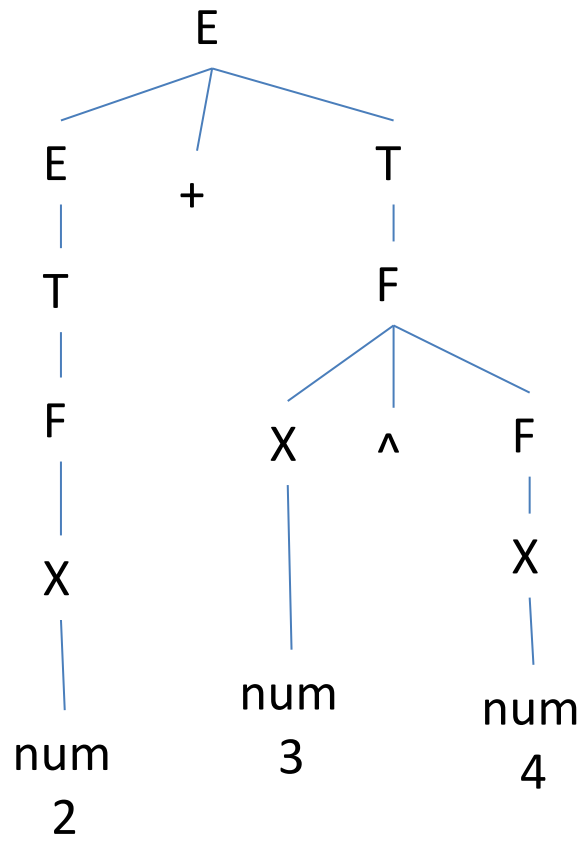
Parse tree???

For input, 2 + 3 ^ 4

SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

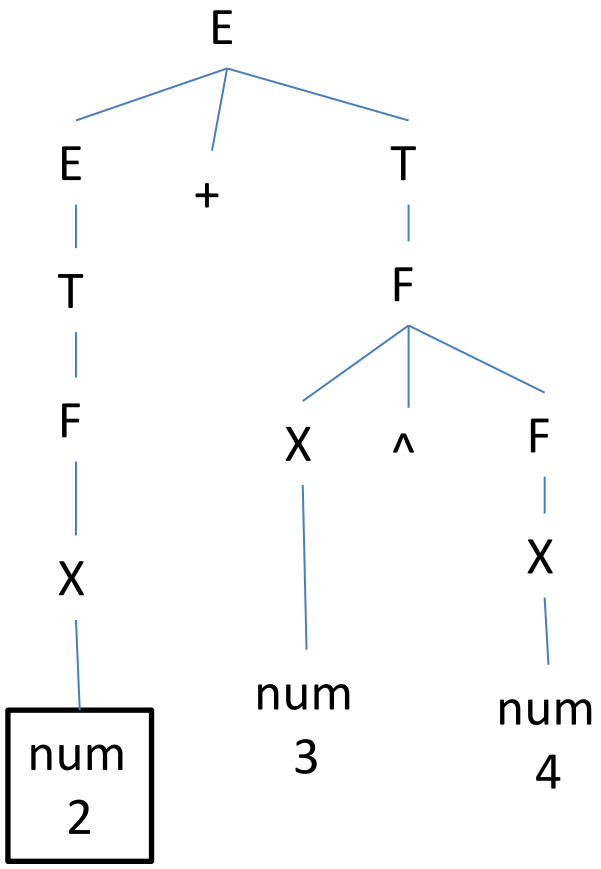
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

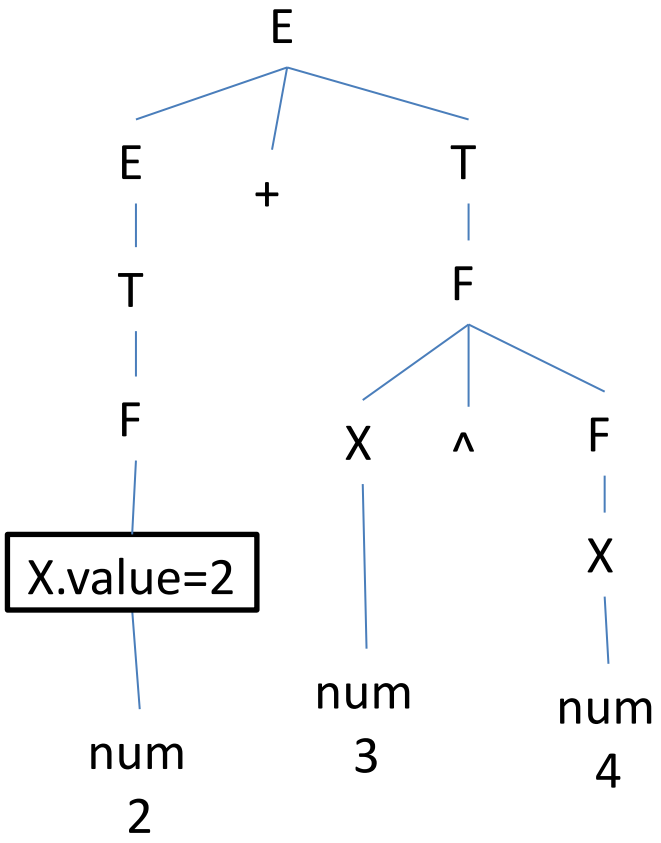
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

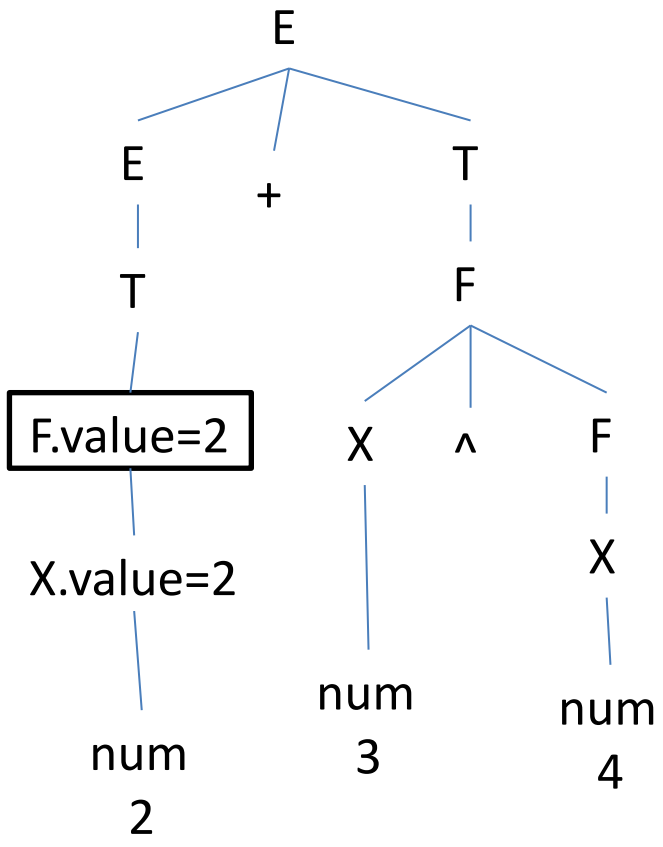
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

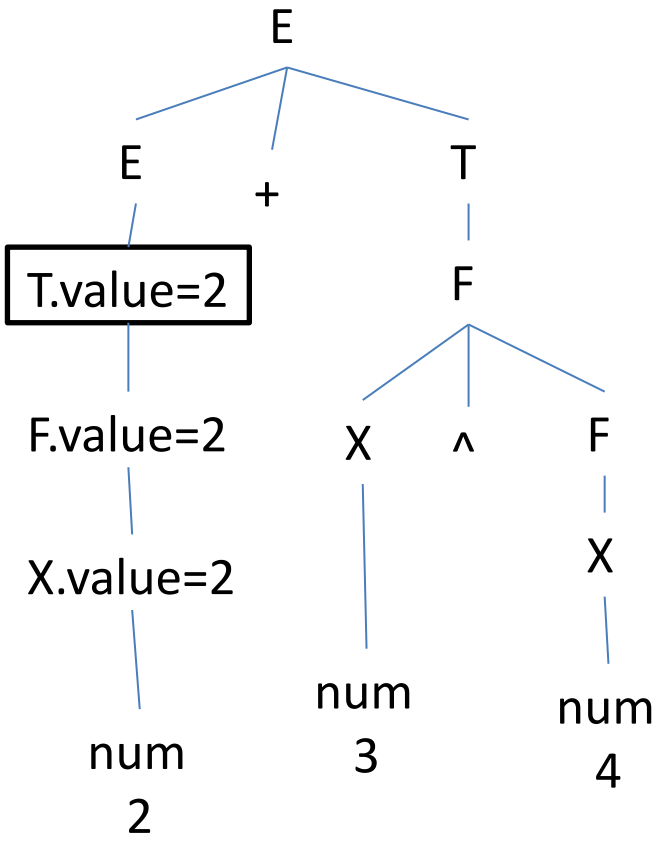
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

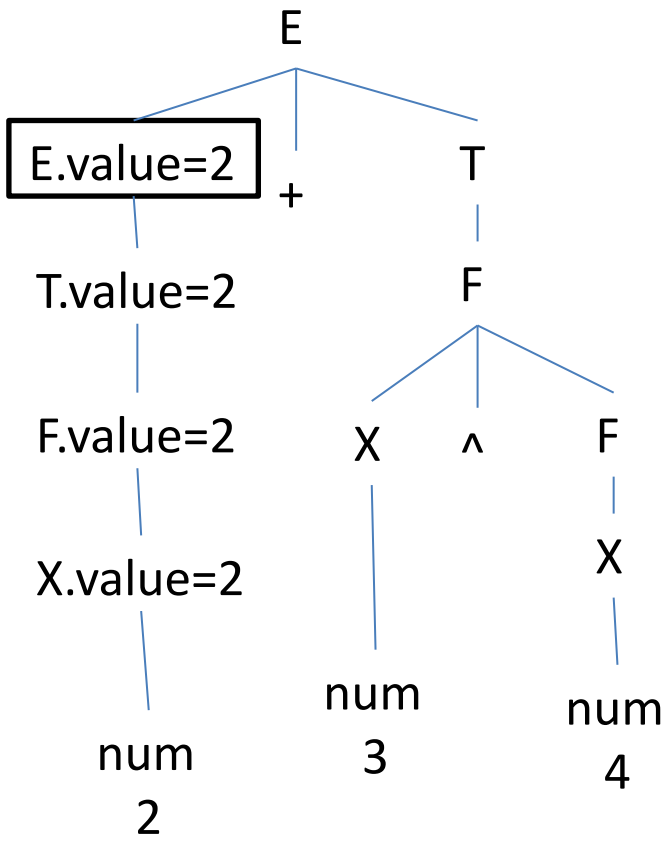
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

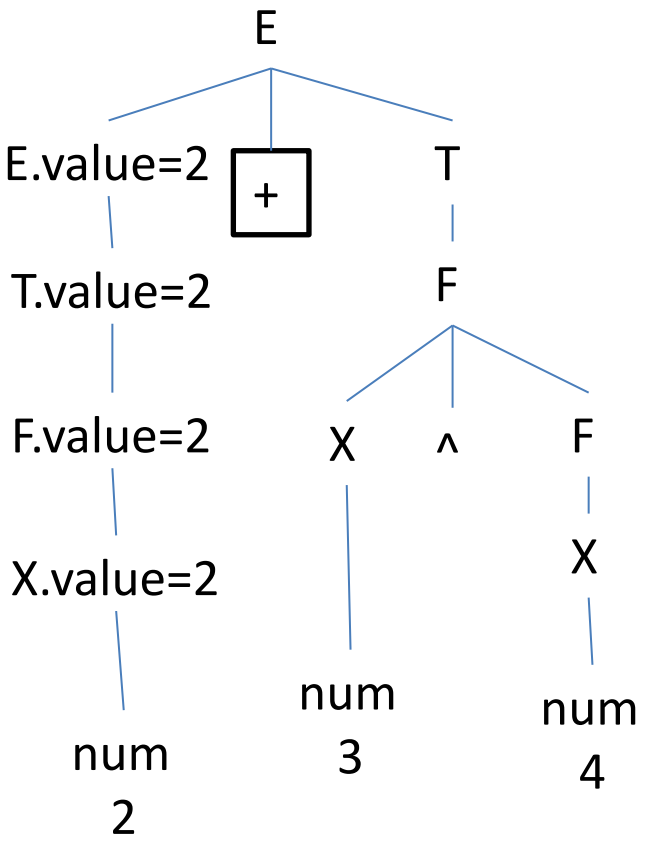
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

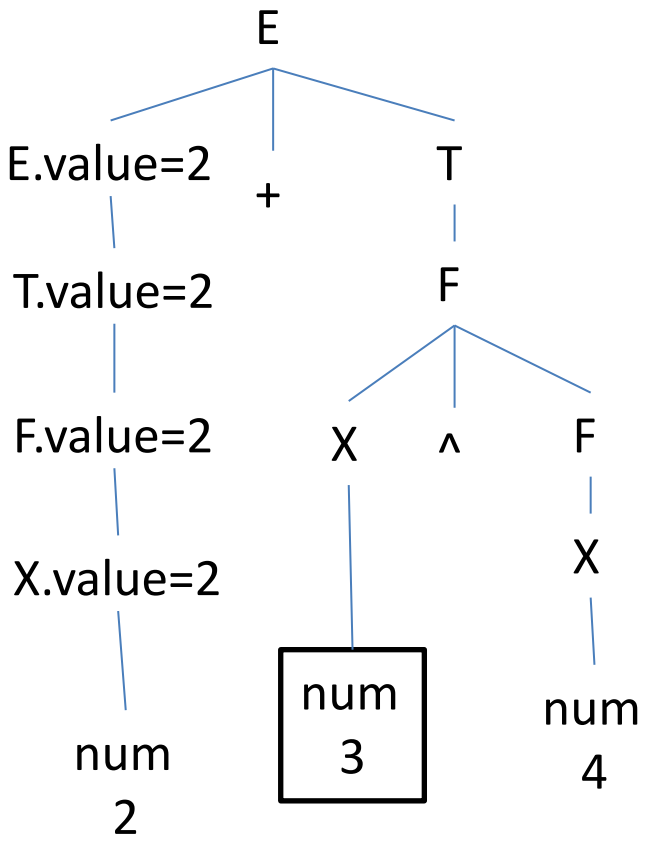
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

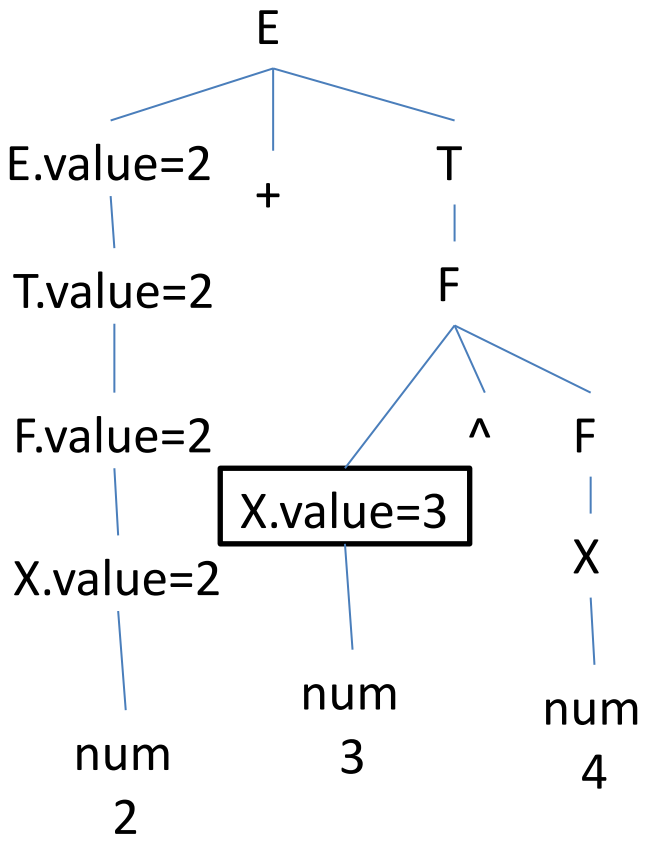
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

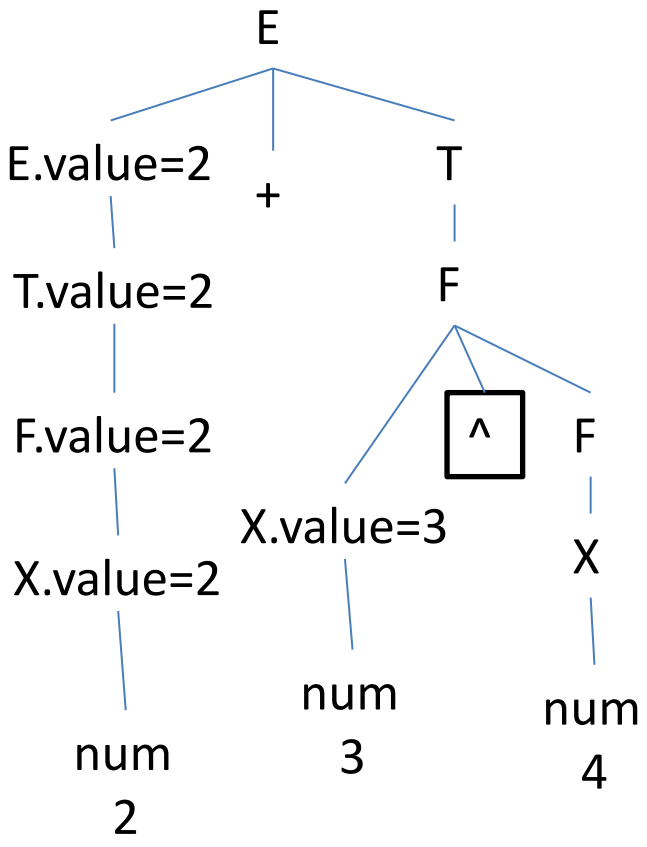
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

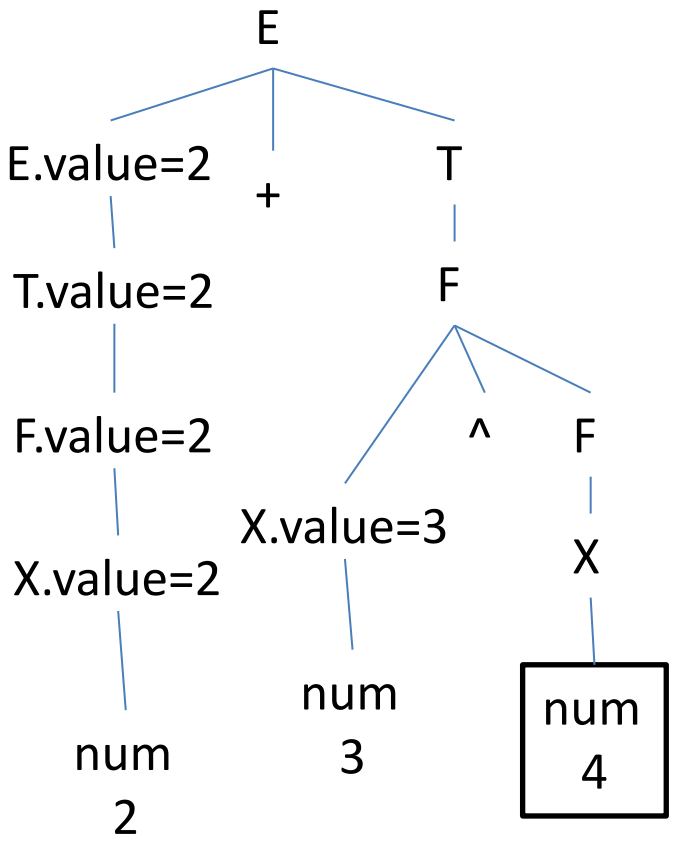
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

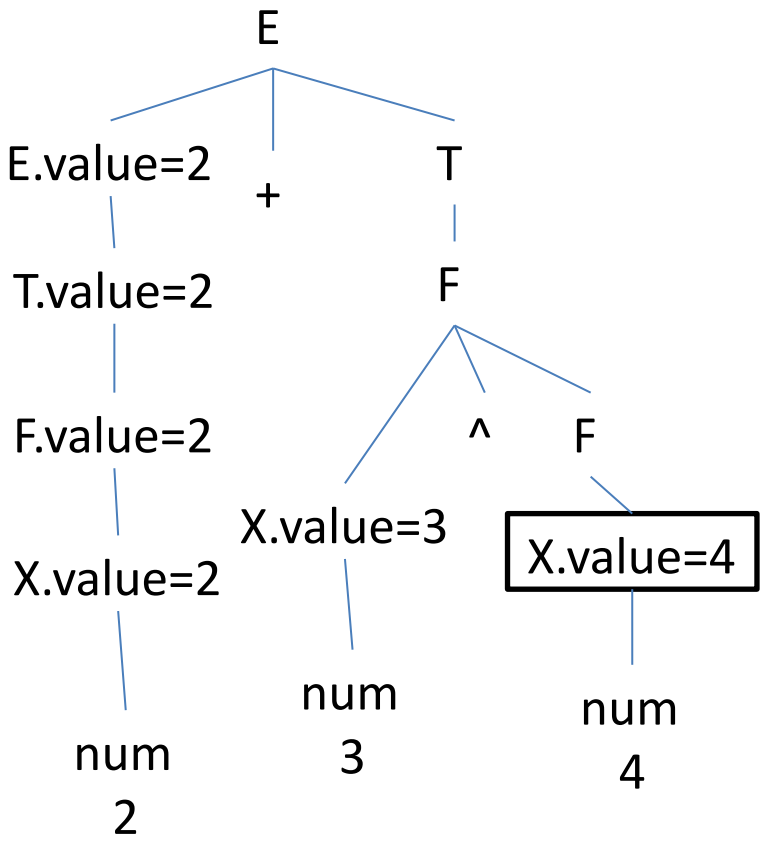
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

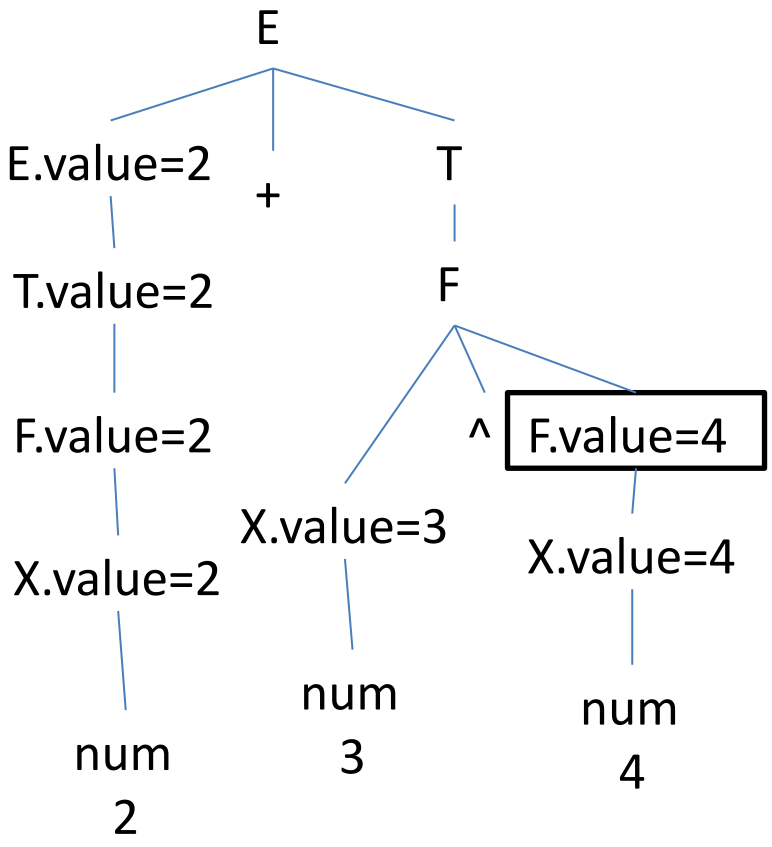
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

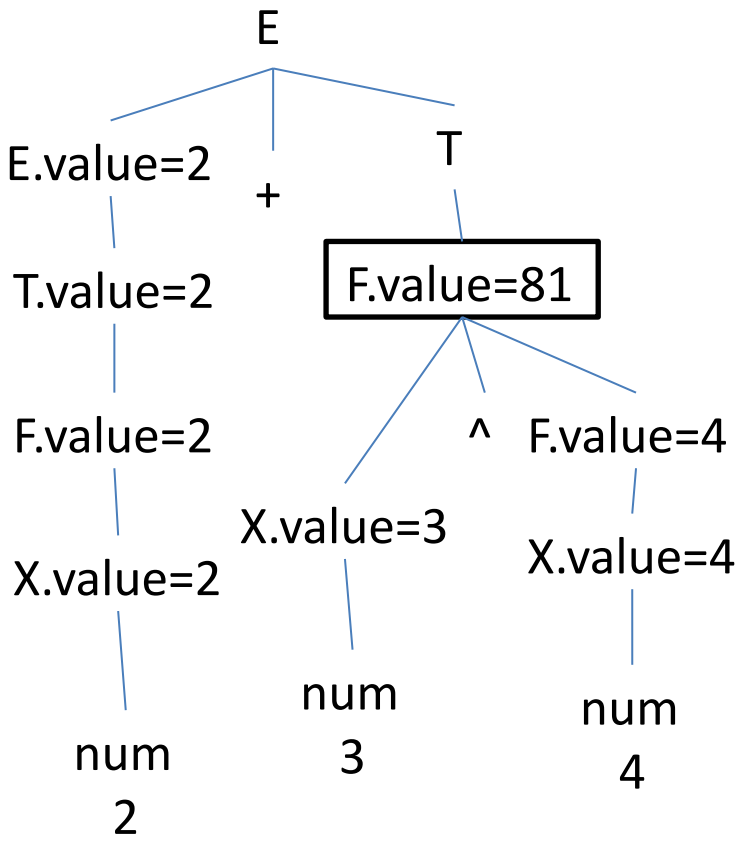
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

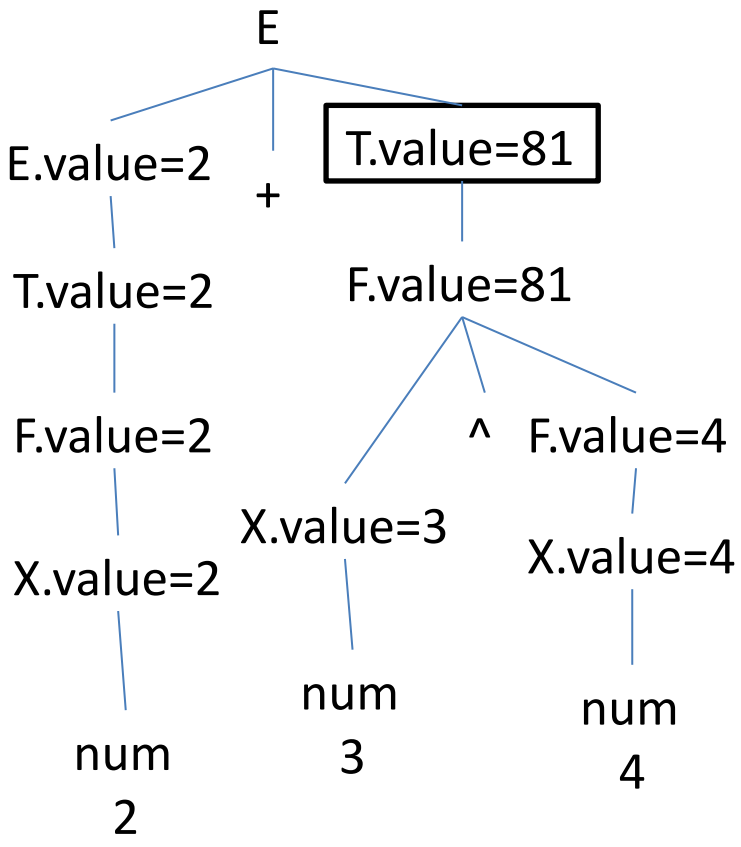
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

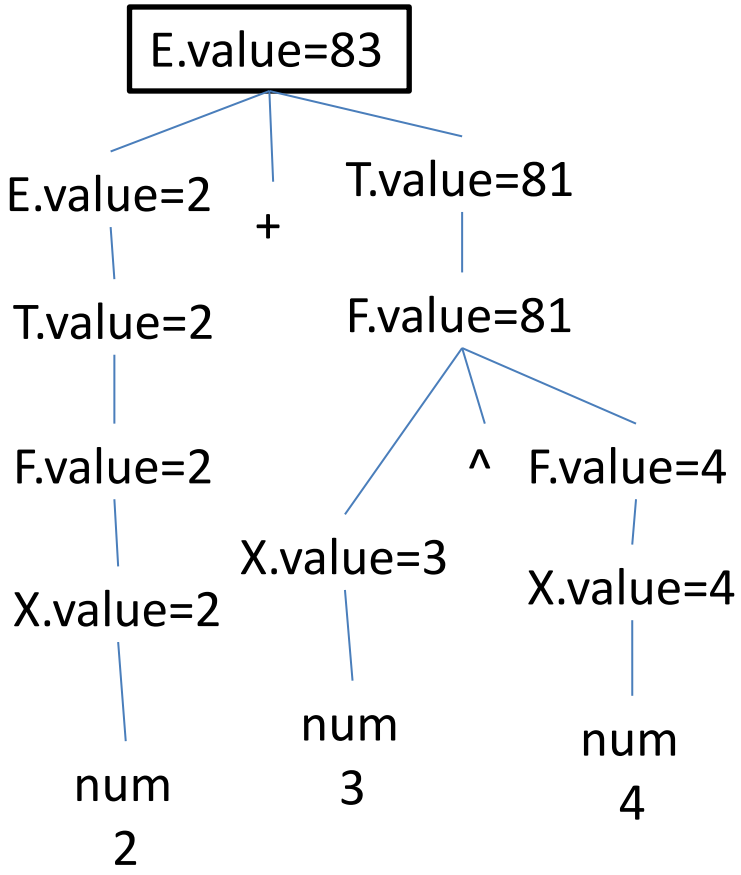
For input, 2 + 3 ^ 4



SDT for desktop calculator

$E \rightarrow E_1 + T$	$\{E.value = E_1.value + T.value\}$
$E \rightarrow T$	$\{E.value = T.value\}$
$T \rightarrow T_1 * F$	$\{T.value = T_1.value * F.value\}$
$T \rightarrow F$	$\{T.value = F.value\}$
$F \rightarrow X \wedge F_1$	$\{F.value = X.value \wedge F_1.value\}$
$F \rightarrow X$	$\{F.value = X.value\}$
$X \rightarrow num$	$\{X.value = num.lexvalue\}$

For input, 2 + 3 ^ 4
Output: 83



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

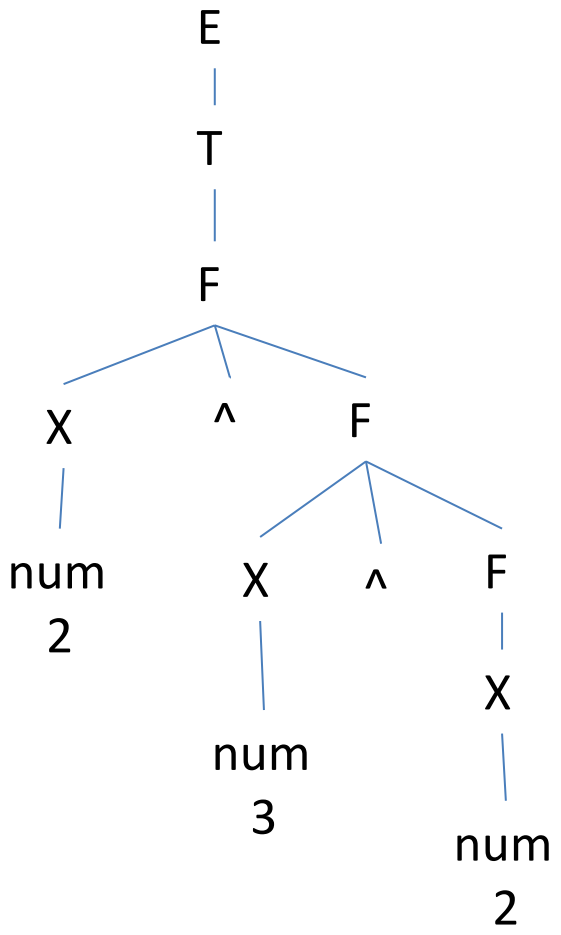
Parse tree???

For input, 2 ^ 3 ^ 2

SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

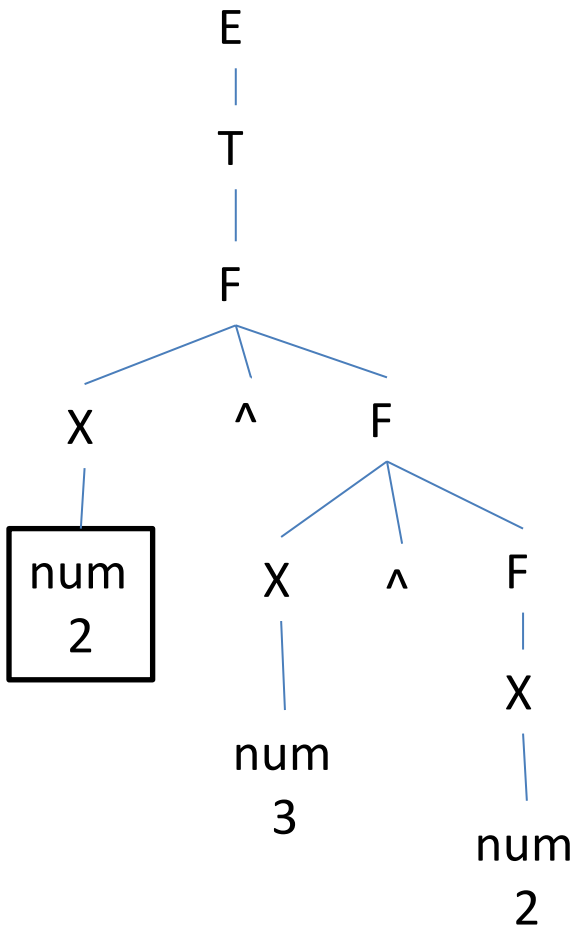
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

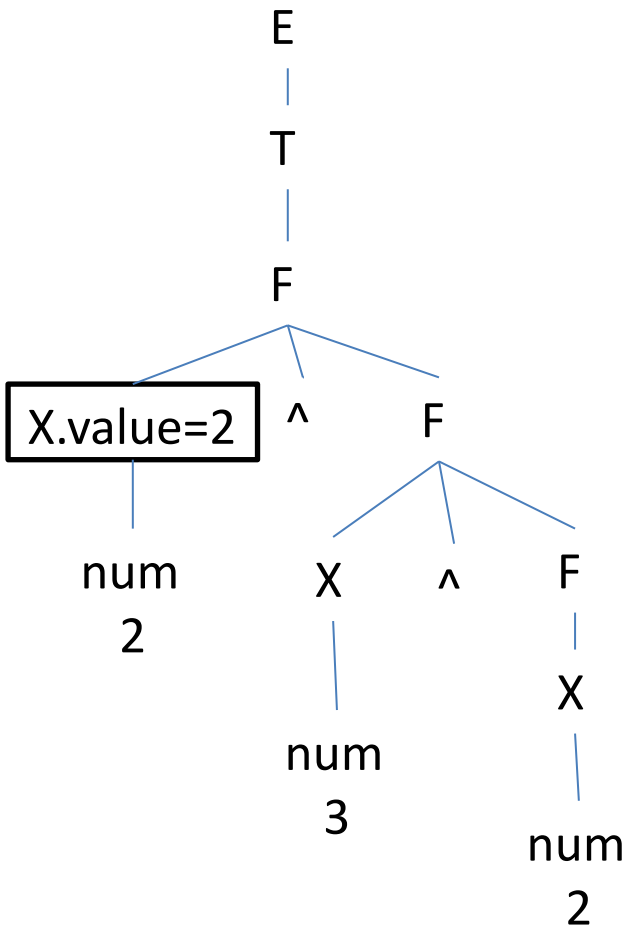
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

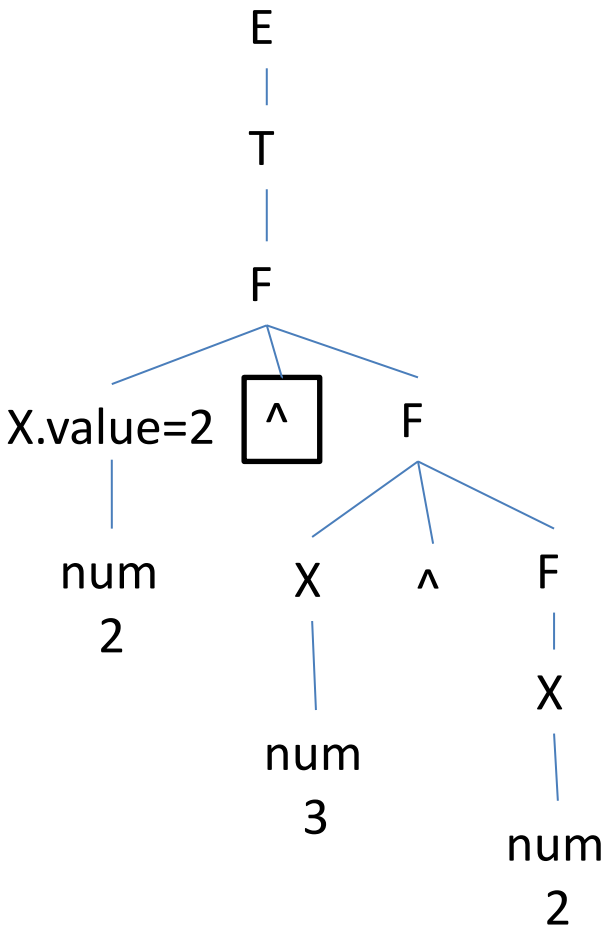
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

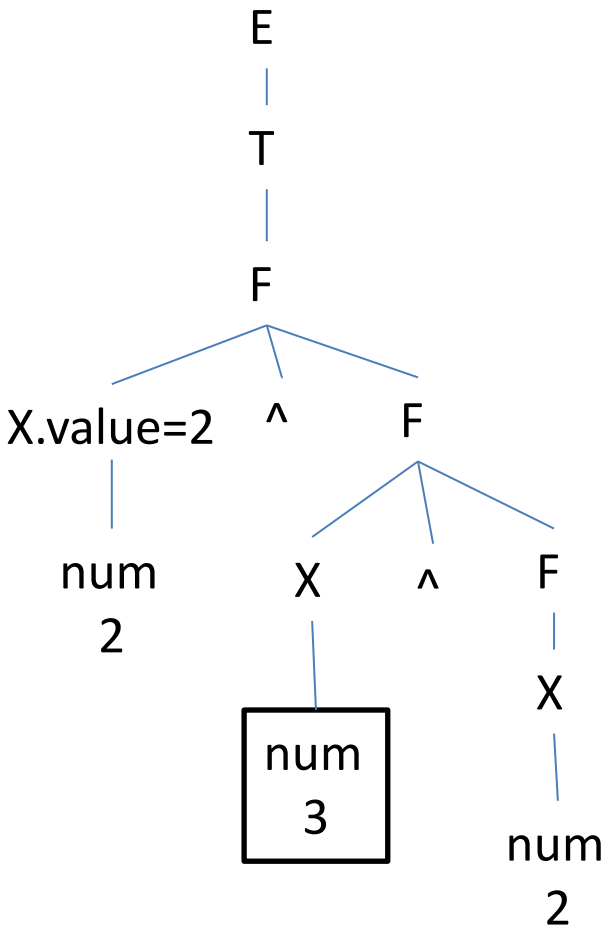
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

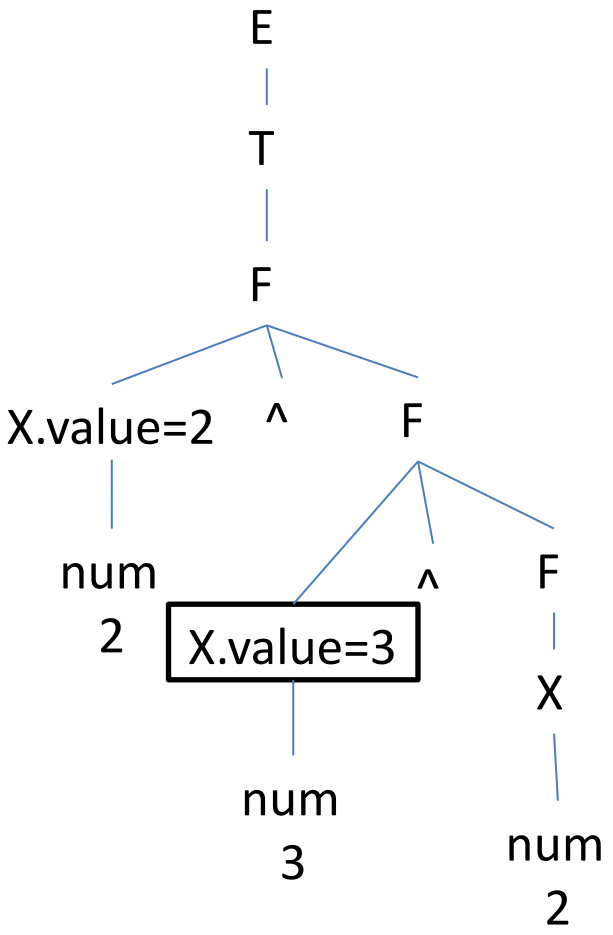
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

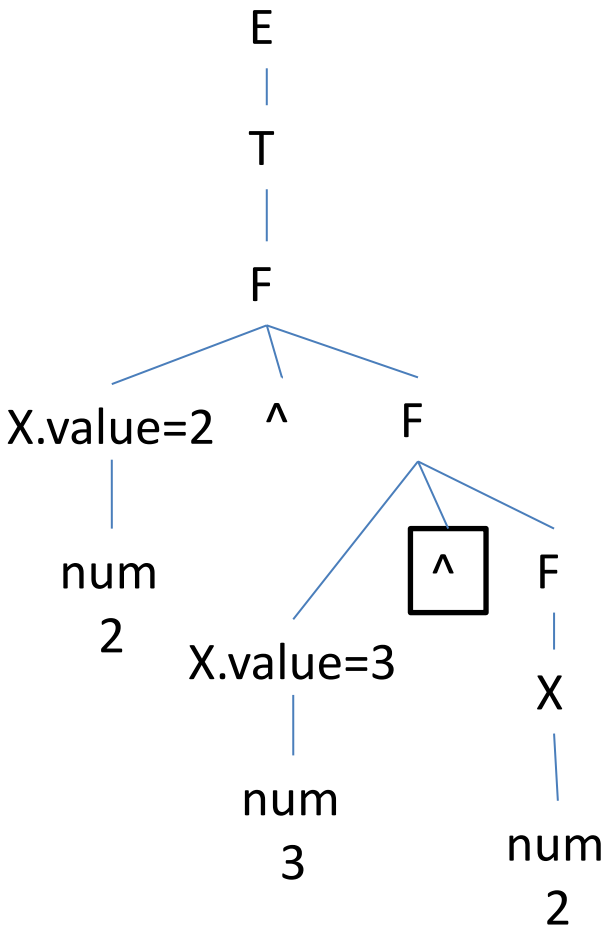
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

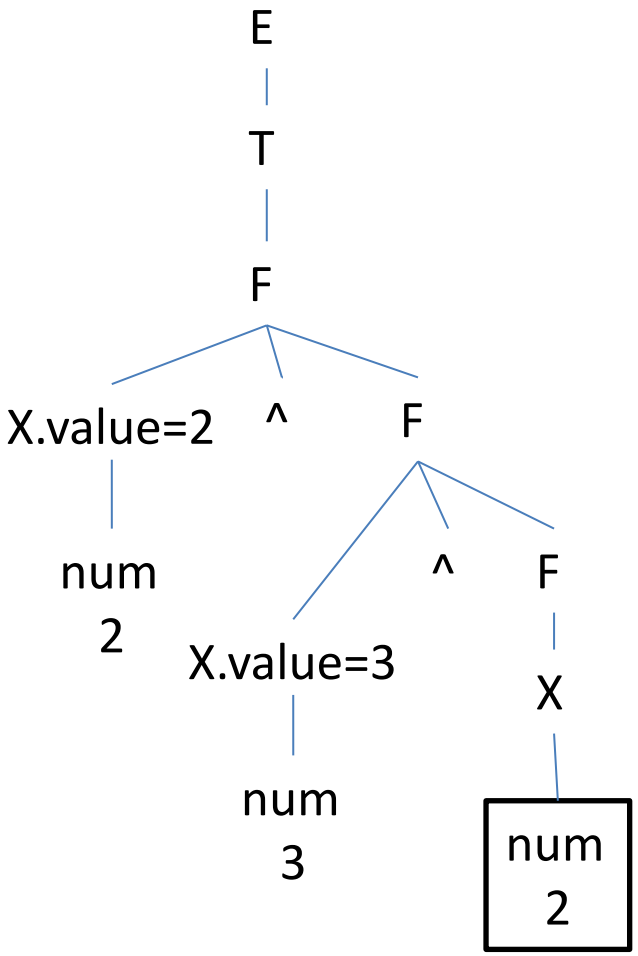
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

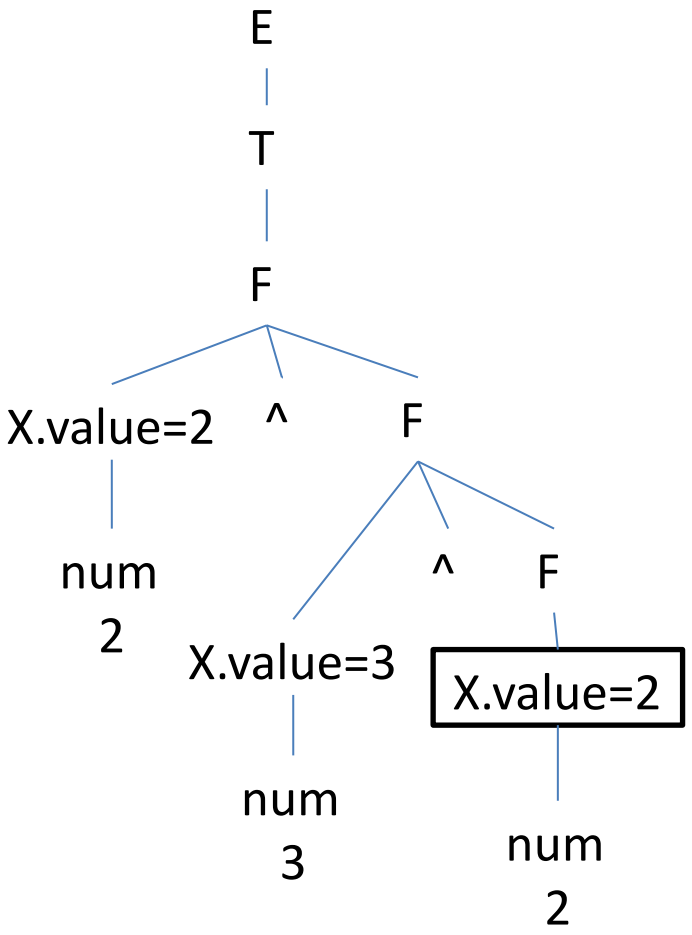
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

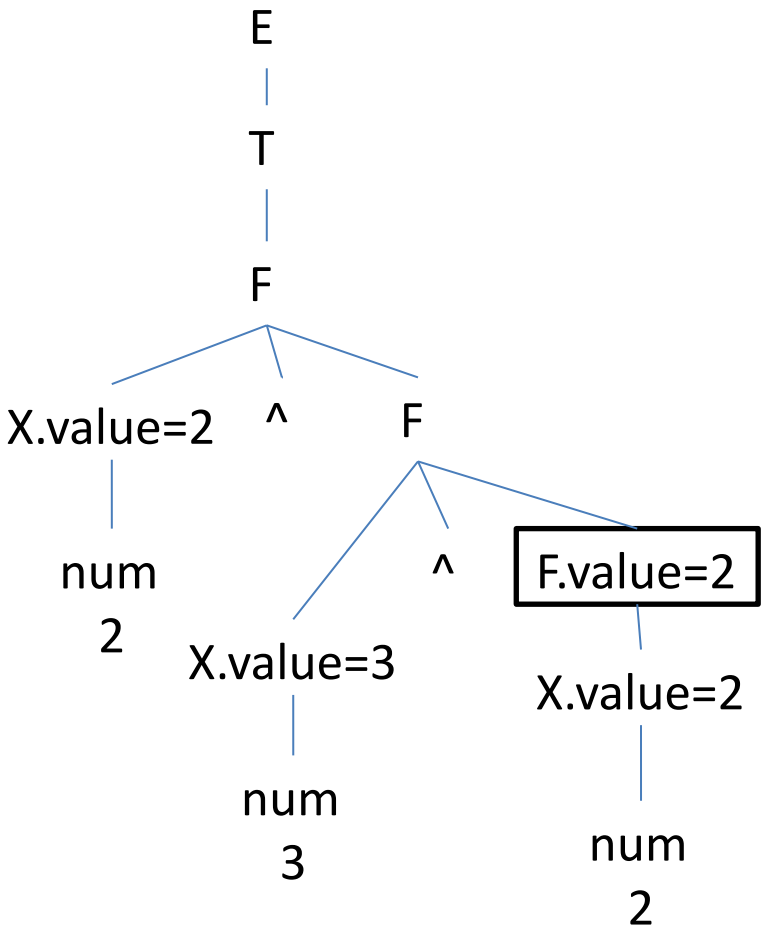
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

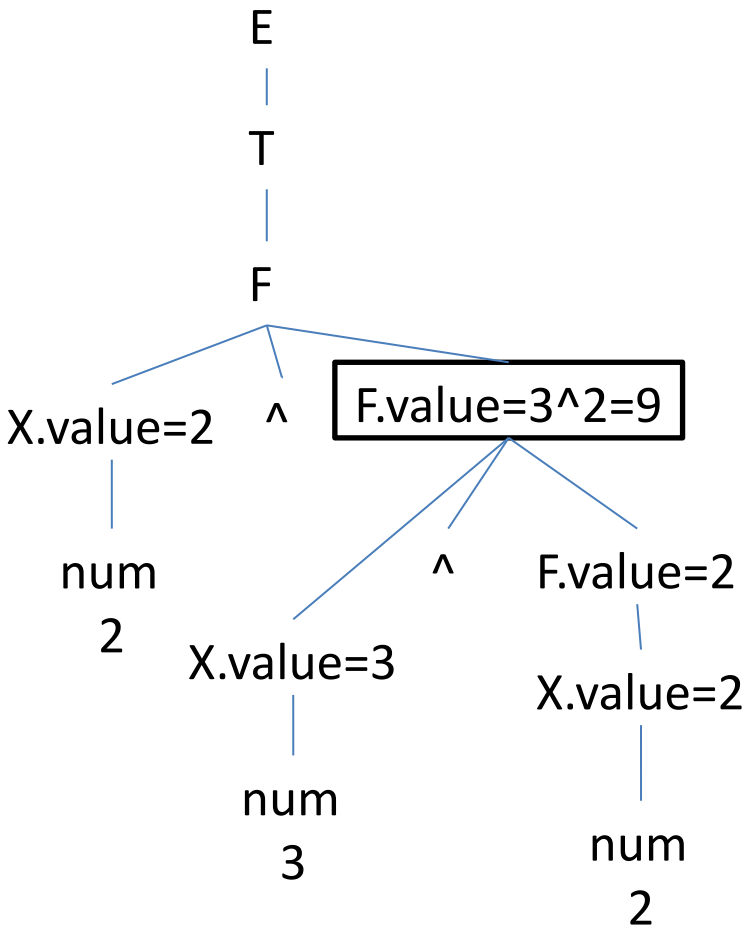
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

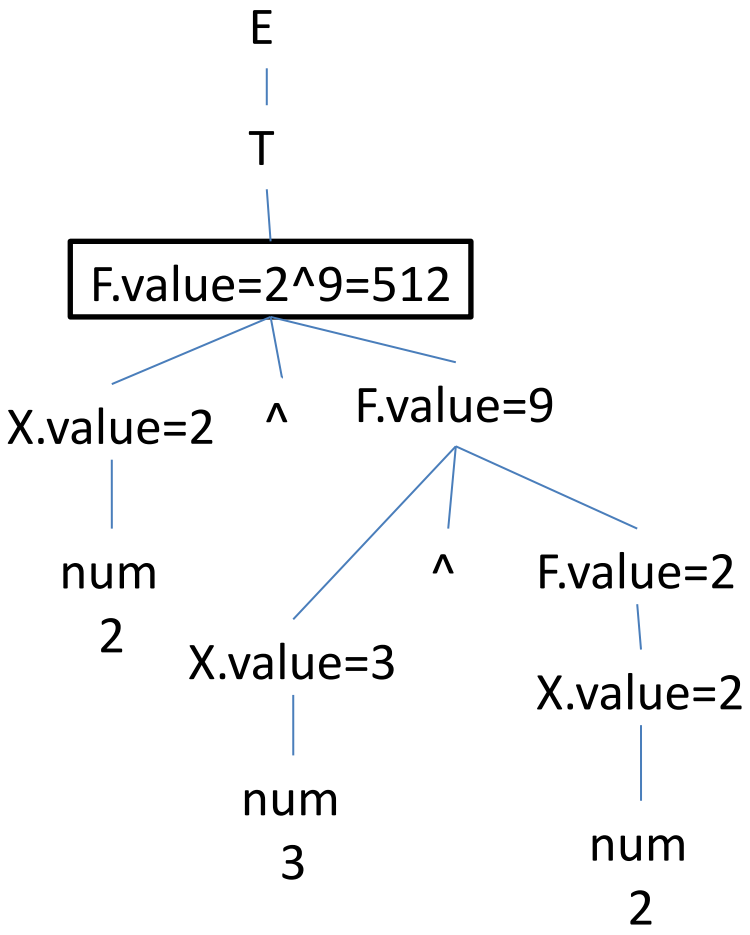
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

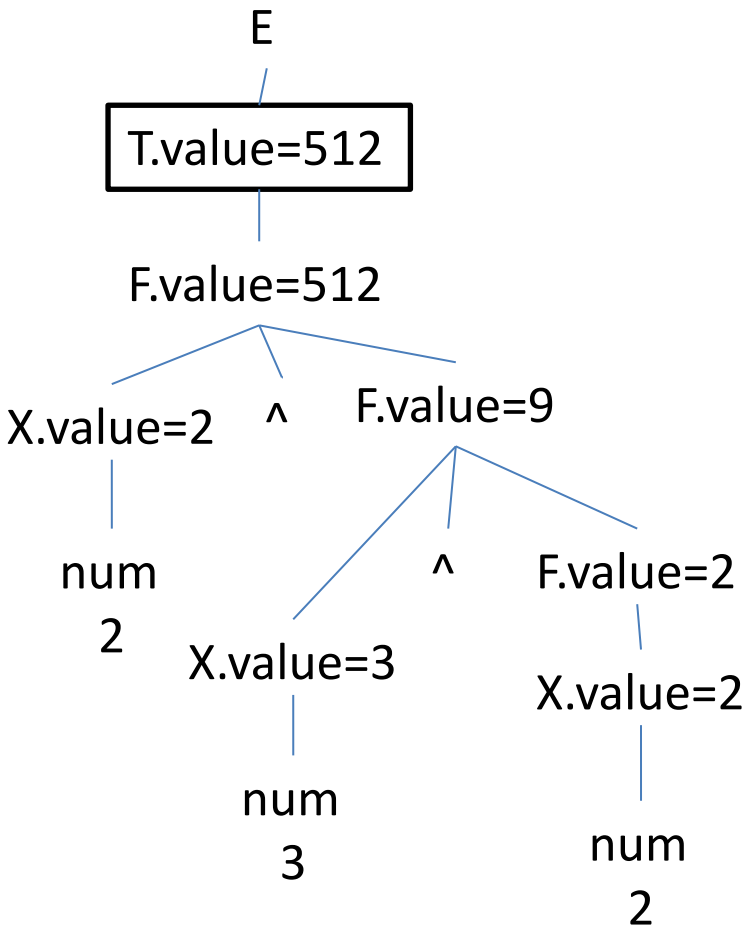
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

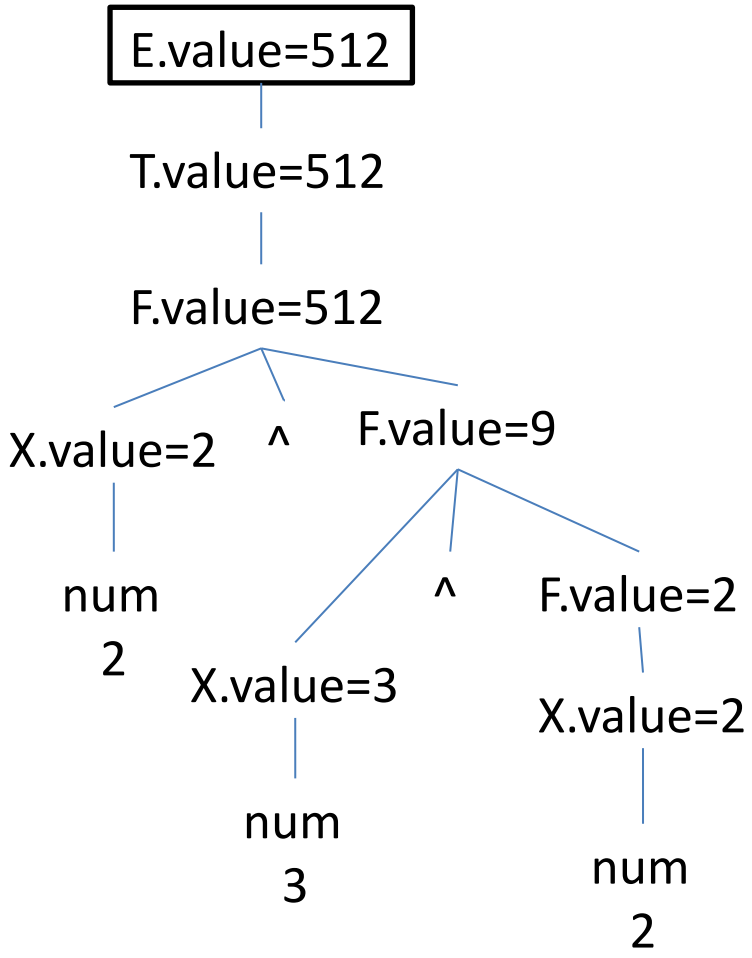
For input, 2 ^ 3 ^ 2



SDT for desktop calculator

$E \rightarrow E_1 + T$	{E.value = E_1.value + T.value}
$E \rightarrow T$	{E.value = T.value}
$T \rightarrow T_1 * F$	{T.value = T_1.value * F.value}
$T \rightarrow F$	{T.value = F.value}
$F \rightarrow X \wedge F_1$	{F.value = X.value ^ F_1.value}
$F \rightarrow X$	{F.value = X.value}
$X \rightarrow num$	{X.value = num.lexvalue}

For input, 2 ^ 3 ^ 2
Output: 512



Example 5

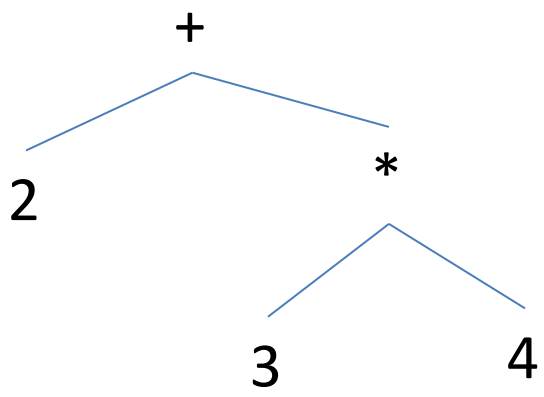
- SDT to build a syntax tree

For input, 2 + 3 * 4

Example 5

- SDT to build a syntax tree

For input, 2 + 3 * 4



Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

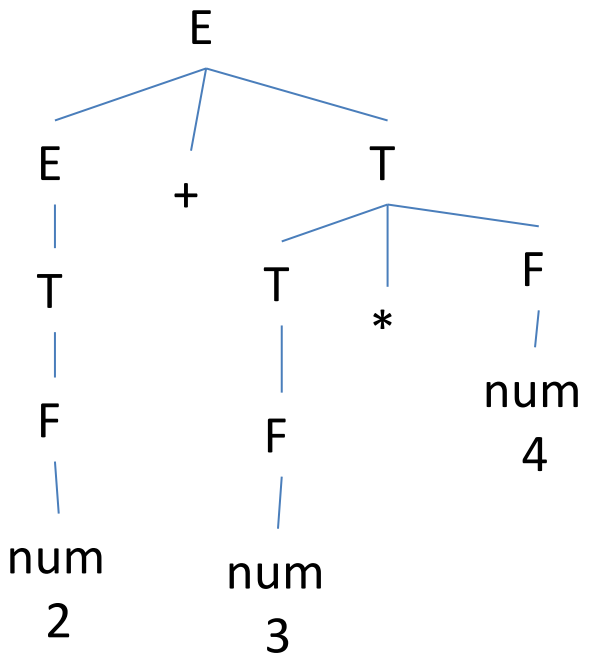
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

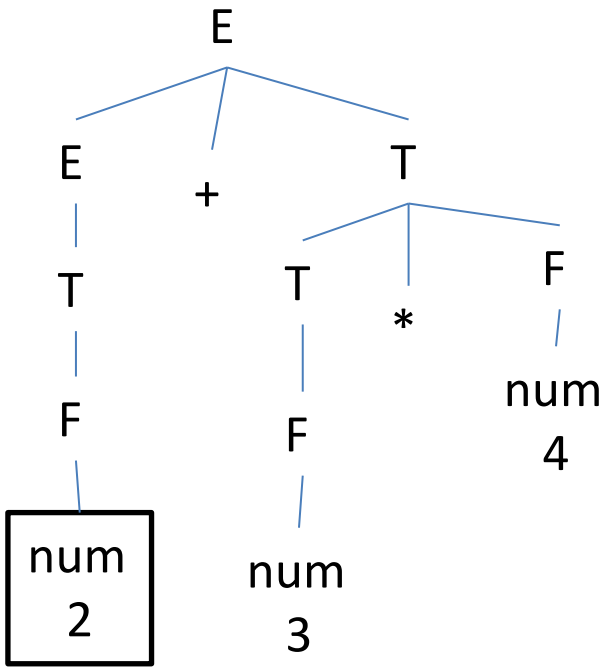
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



Example 5

- SDT to build a syntax tree

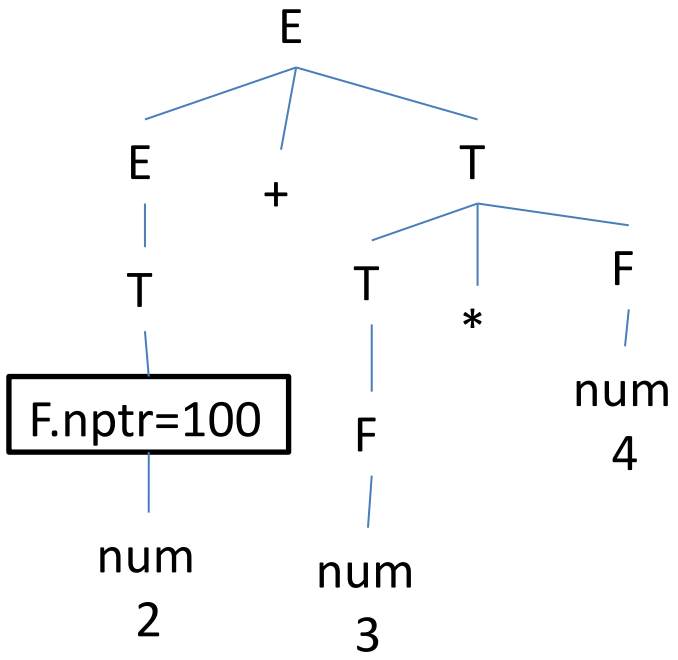
E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}



For input, 2 + 3 * 4

100

null	2	null

Example 5

- SDT to build a syntax tree

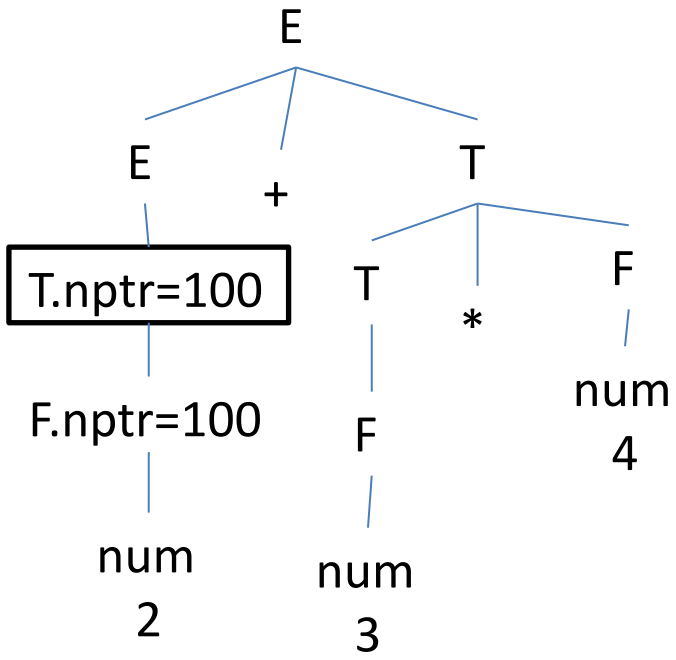
E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}



For input, 2 + 3 * 4

100

null	2	null

Example 5

- SDT to build a syntax tree

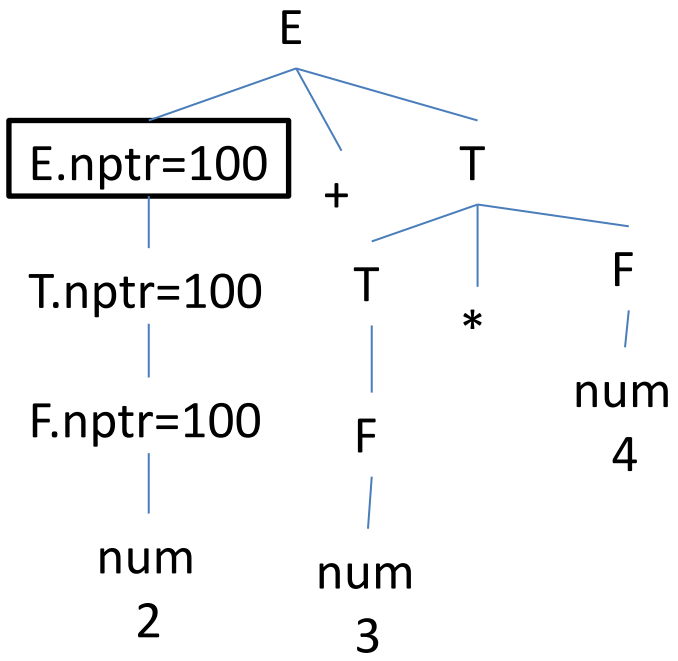
E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}



For input, 2 + 3 * 4

100

null	2	null

Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

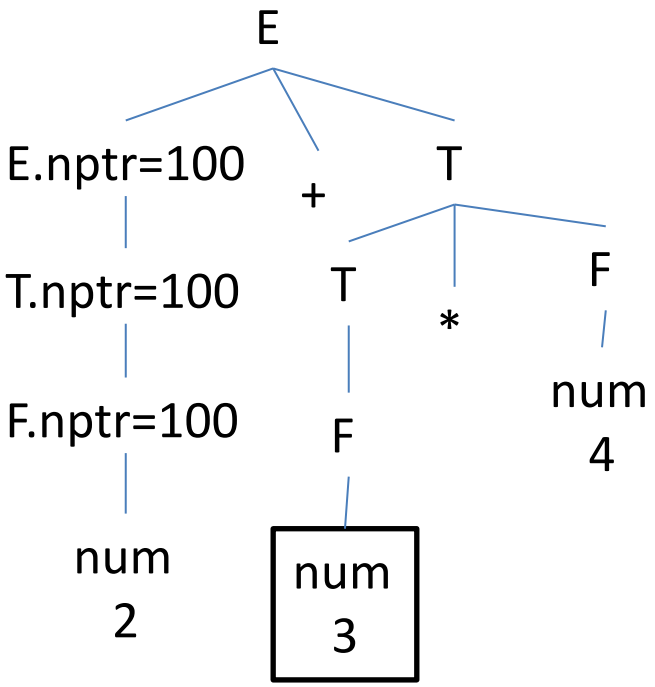
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



100

null	2	null

Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

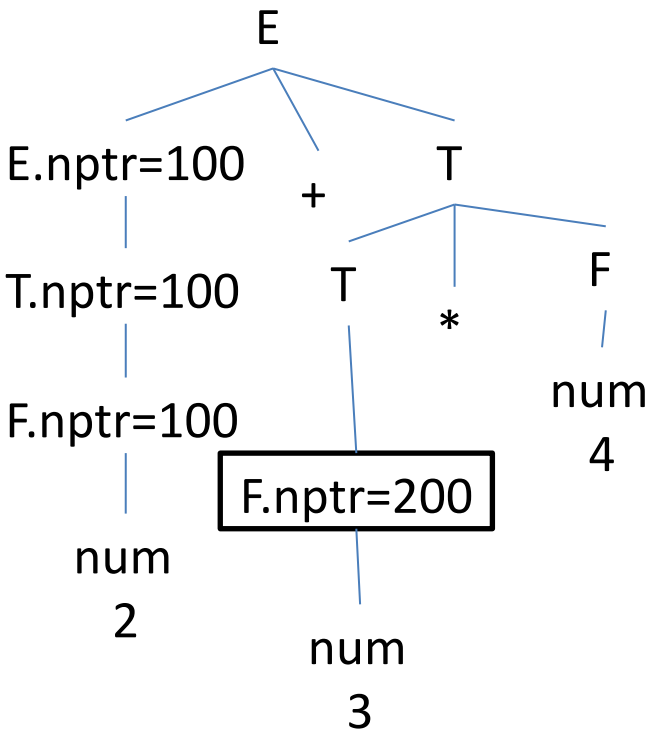
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



100

null	2	null

200

null	3	null

Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

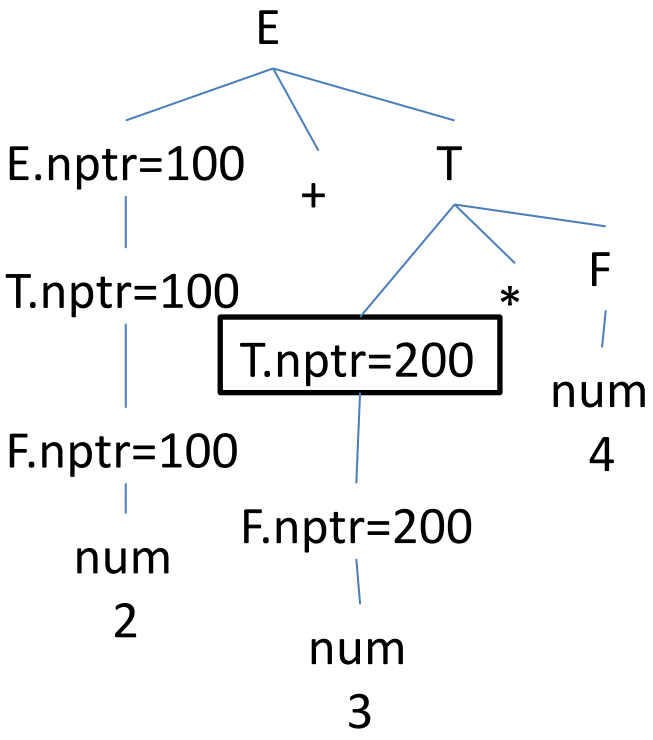
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



100

null	2	null

200

null	3	null

Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

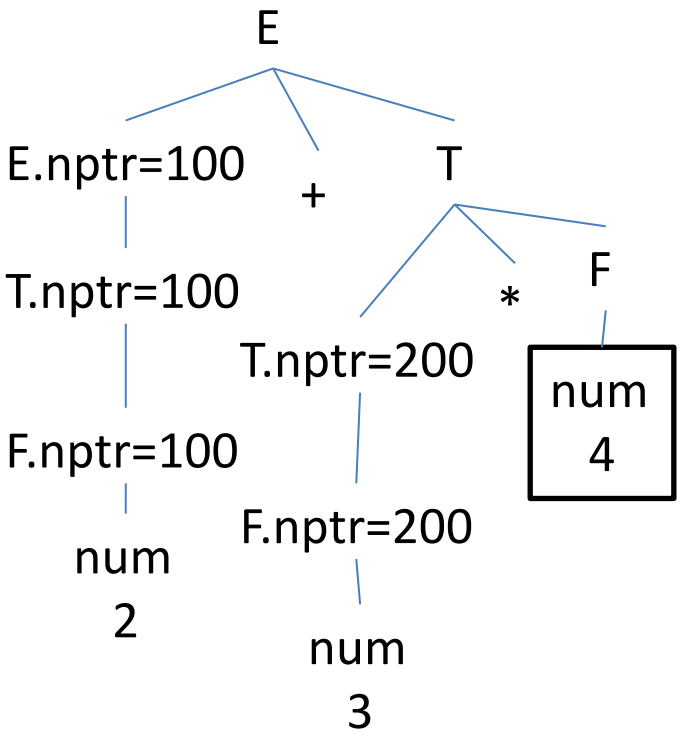
E → T {E.nptr=T.nptr;}

T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4



100

null	2	null

200

null	3	null

Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

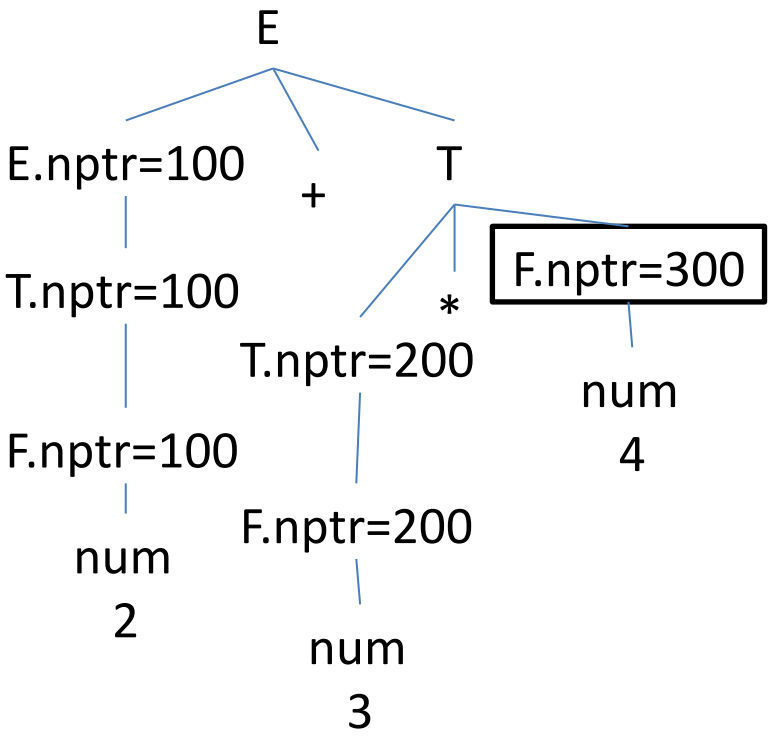
E → T {E.nptr=T.nptr;}

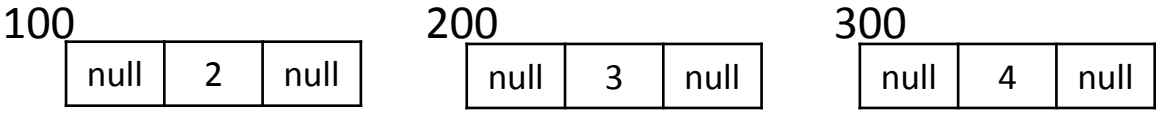
T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4





Example 5

- SDT to build a syntax tree

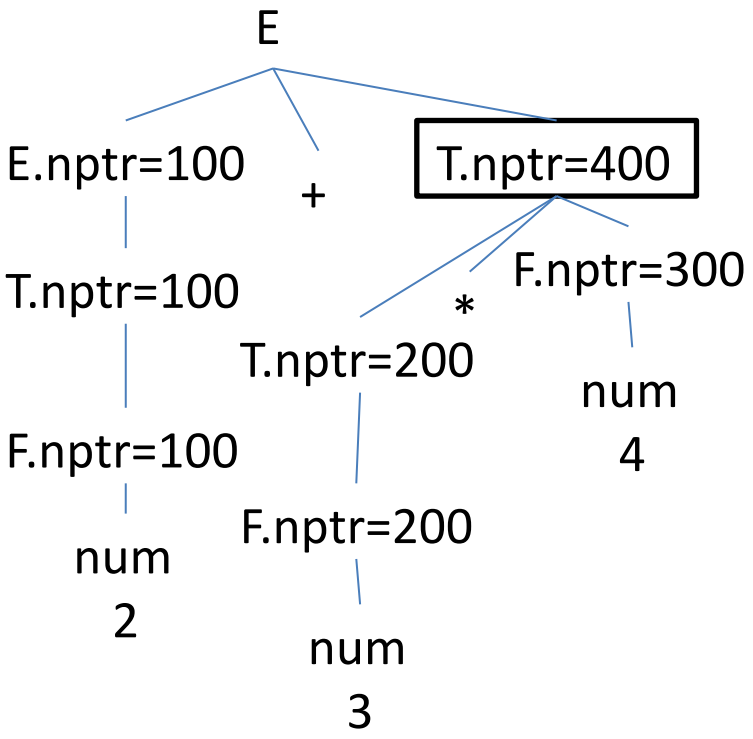
E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

E → T {E.nptr=T.nptr;}

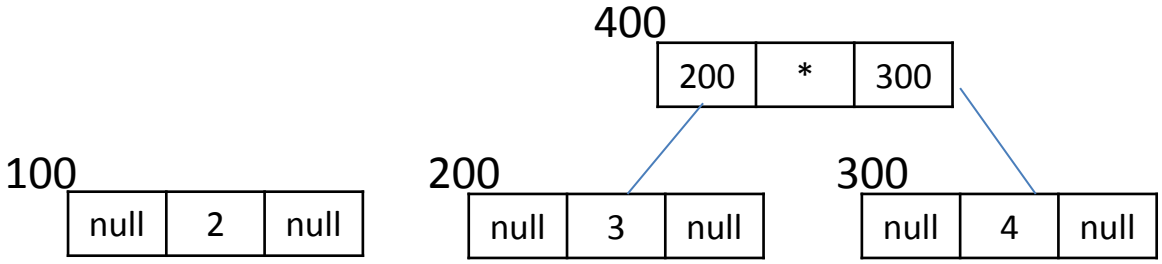
T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}



For input, 2 + 3 * 4



Example 5

- SDT to build a syntax tree

E→ E + T {E.nptr=mknode(E.nptr, '+', T.nptr);}

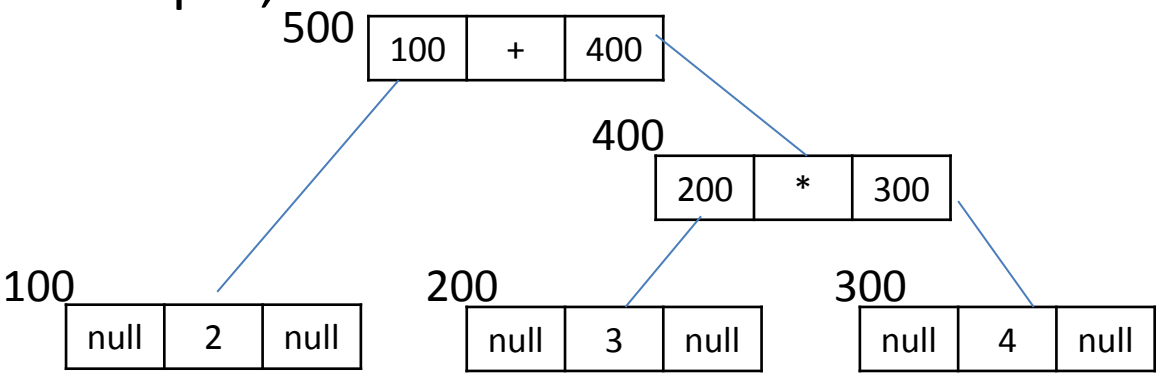
E → T {E.nptr=T.nptr;}

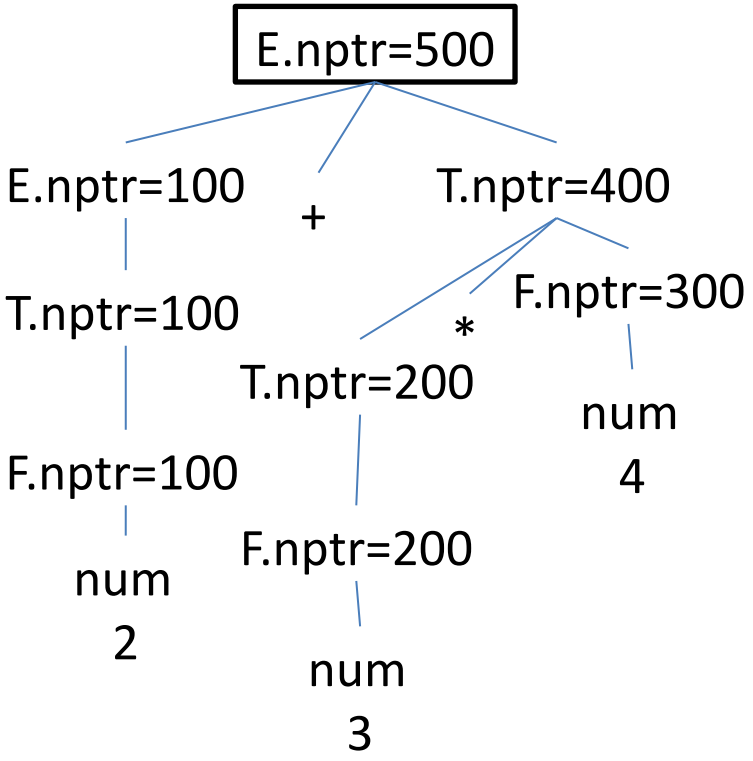
T → T * F {T.nptr=mknode(T.nptr, *, F.nptr);}

T→F {T.nptr=F.nptr}

F→ num {F.nptr=mknode(null, idname, null);}

For input, 2 + 3 * 4





Example 6

- SDT to generate three address code

$S \rightarrow id = E$ {gen(id.name = E.place}

$E \rightarrow E_1 + T$ {E.place = newTemp(); gen(E.place = E_1.place + T.place);}

$E \rightarrow T$ {E.place = T.place}

$T \rightarrow T_1 * F$ {T.place = newTemp(); gen(E.place = T_1.place * F.place);}

$T \rightarrow F$ {T.place = F.place}

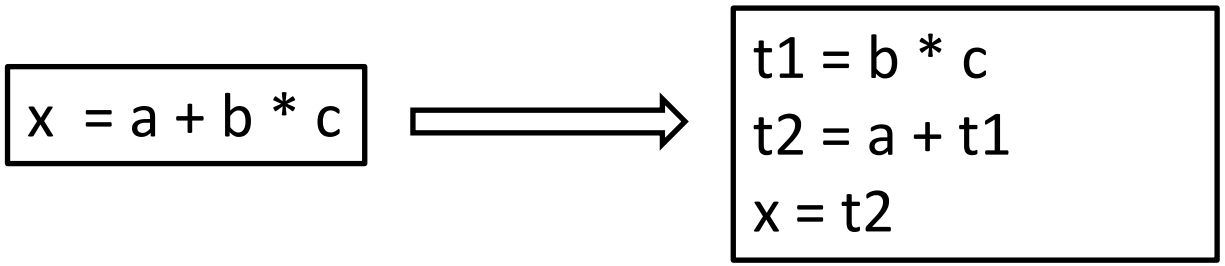
$F \rightarrow id$ {F.place = id.name}

gen() :- generates a statement in three address code

Example 6

- SDT to generate three address code

$S \rightarrow id = E$	{gen(id.name = E.place}
$E \rightarrow E_1 + T$	{E.place = newTemp(); gen(E.place = E_1.place + T.place);}
$E \rightarrow T$	{E.place = T.place}
$T \rightarrow T_1 * F$	{T.place = newTemp(); gen(E.place = T_1.place * F.place);}
$T \rightarrow F$	{T.place = F.place}
$F \rightarrow id$	{F.place = id.name}



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

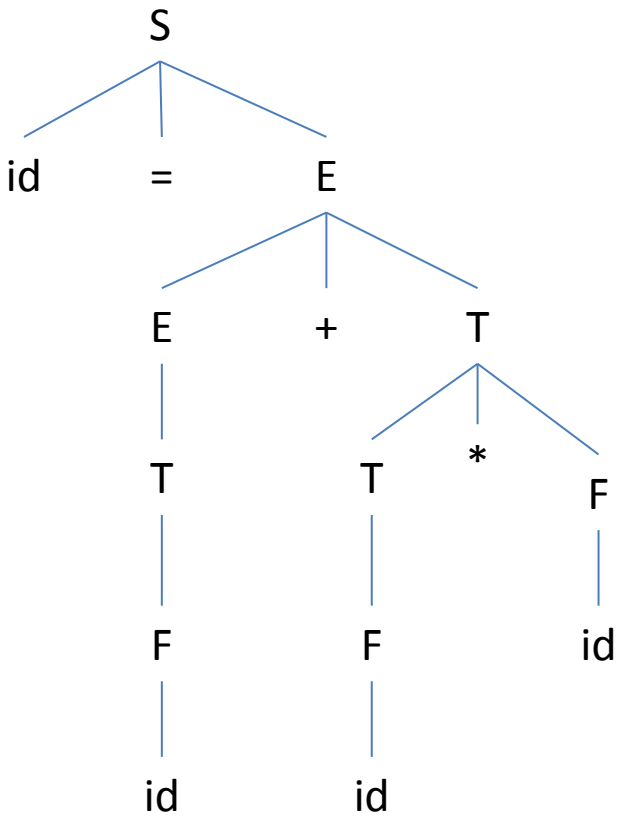
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

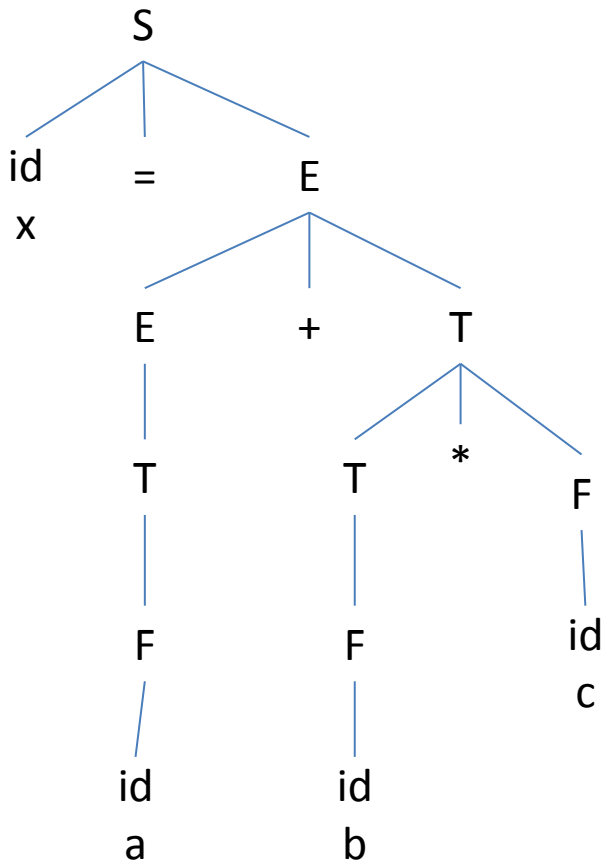
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

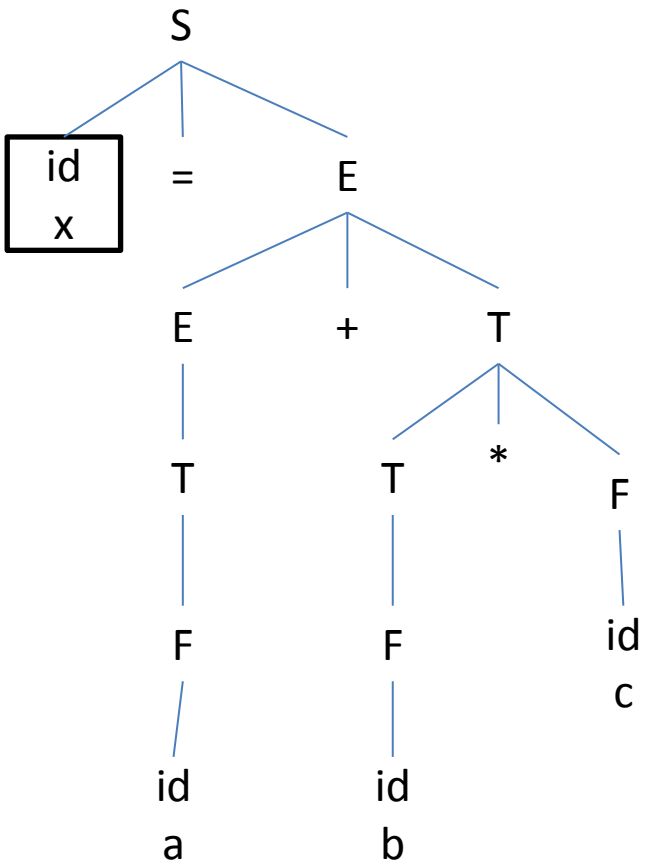
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

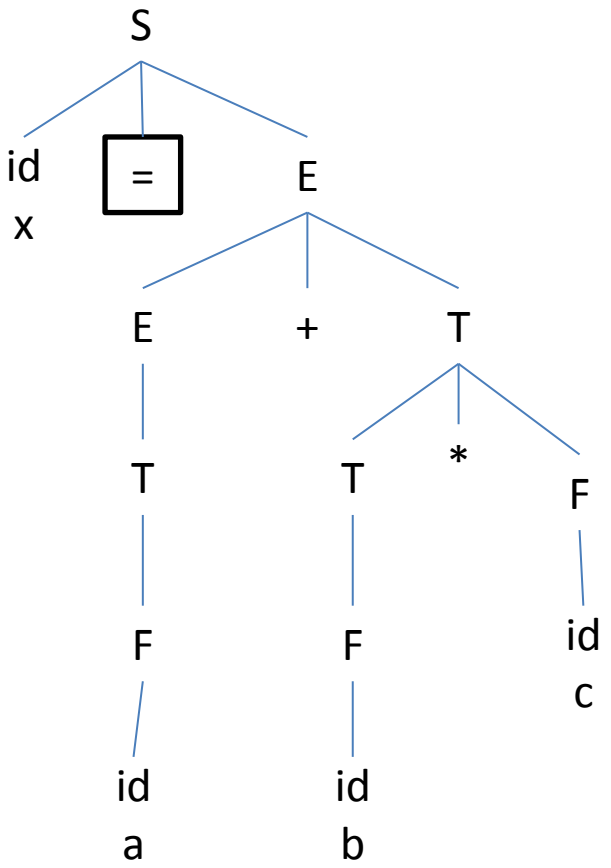
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

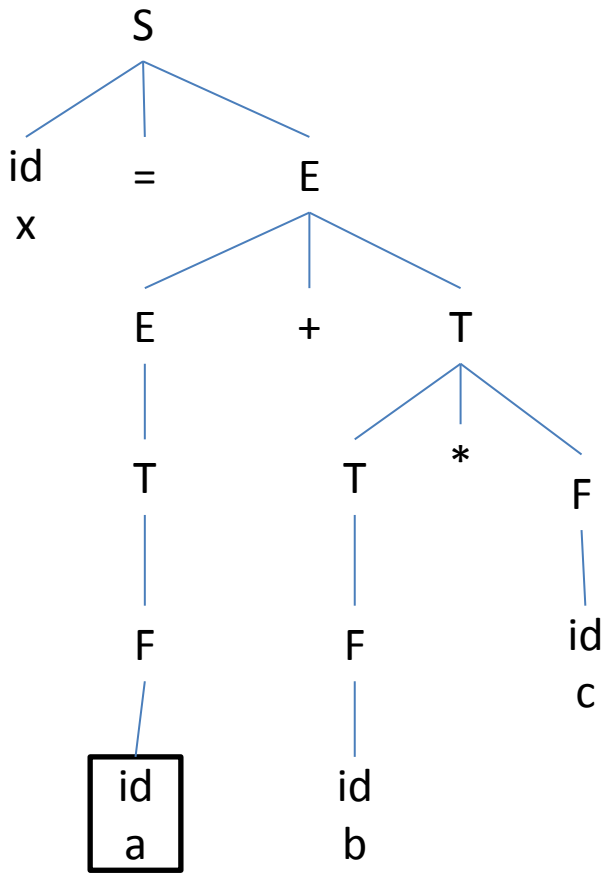
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

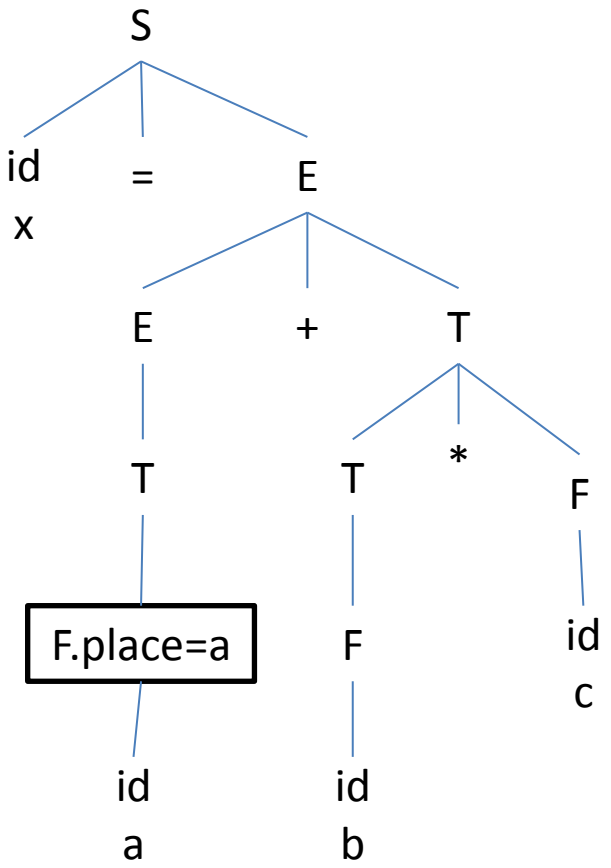
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

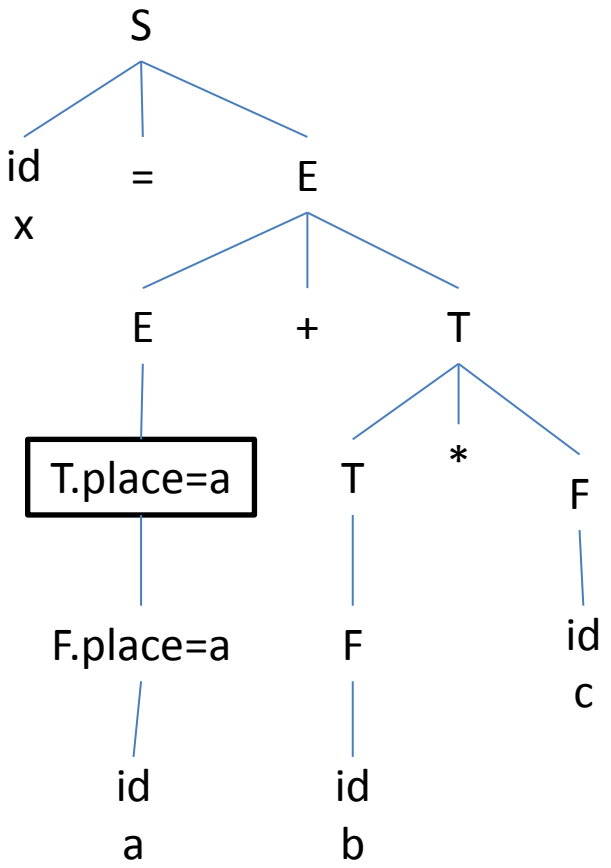
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

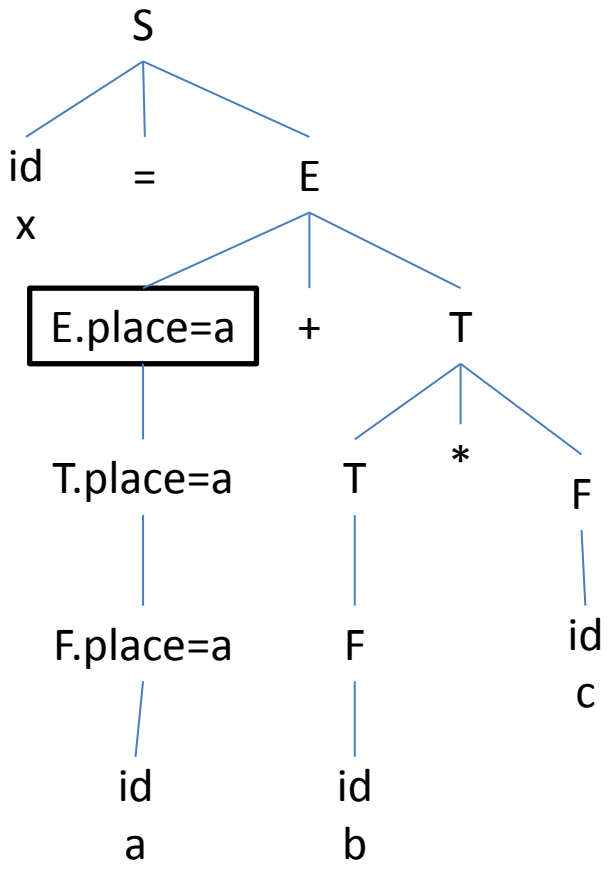
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

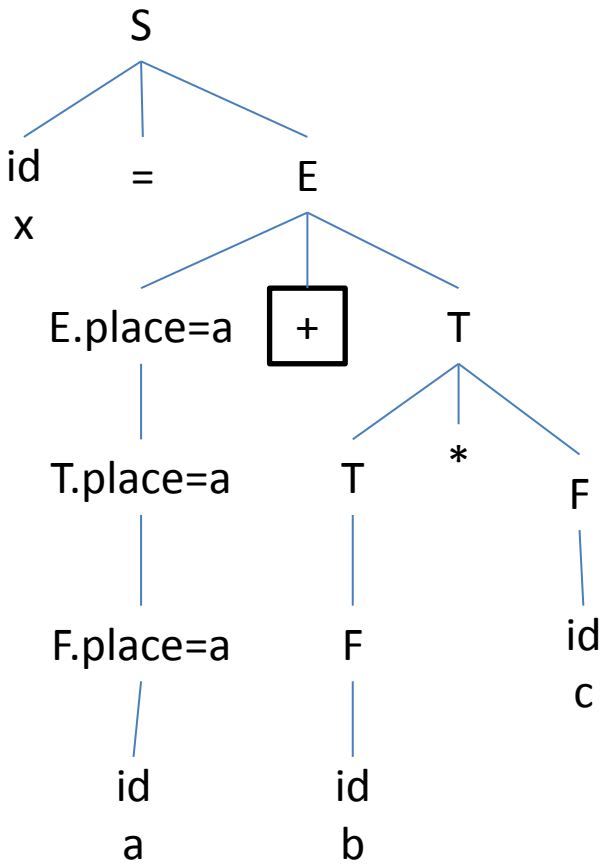
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

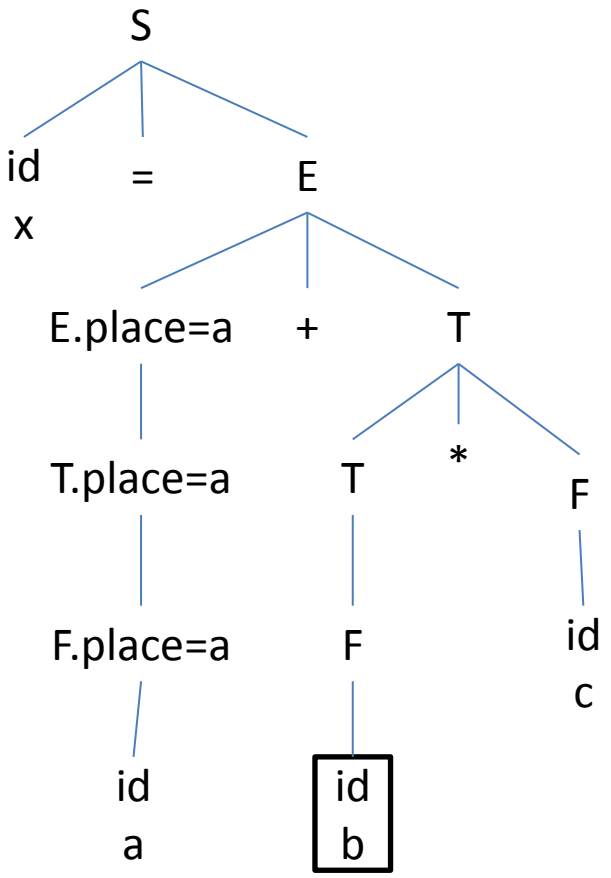
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

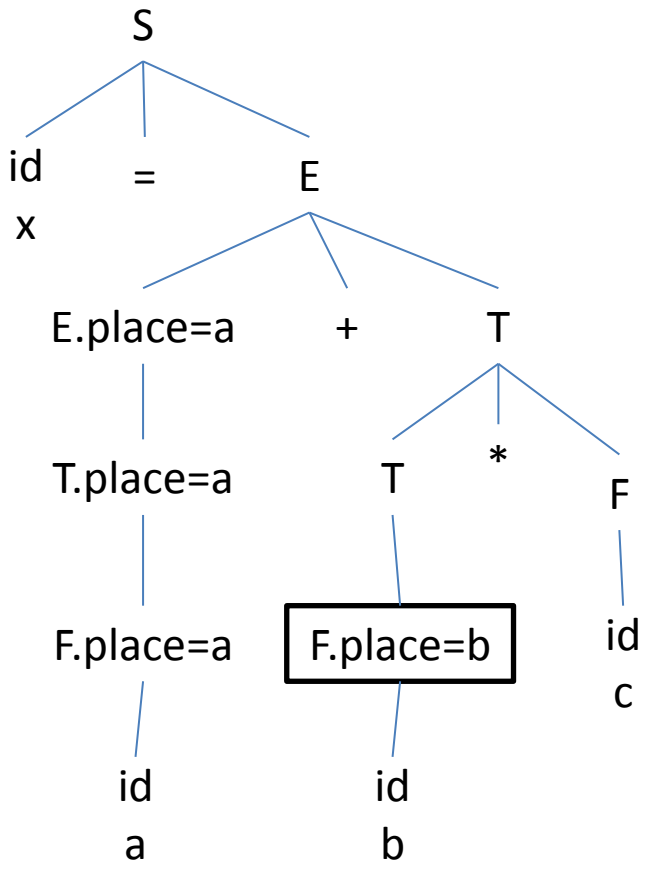
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

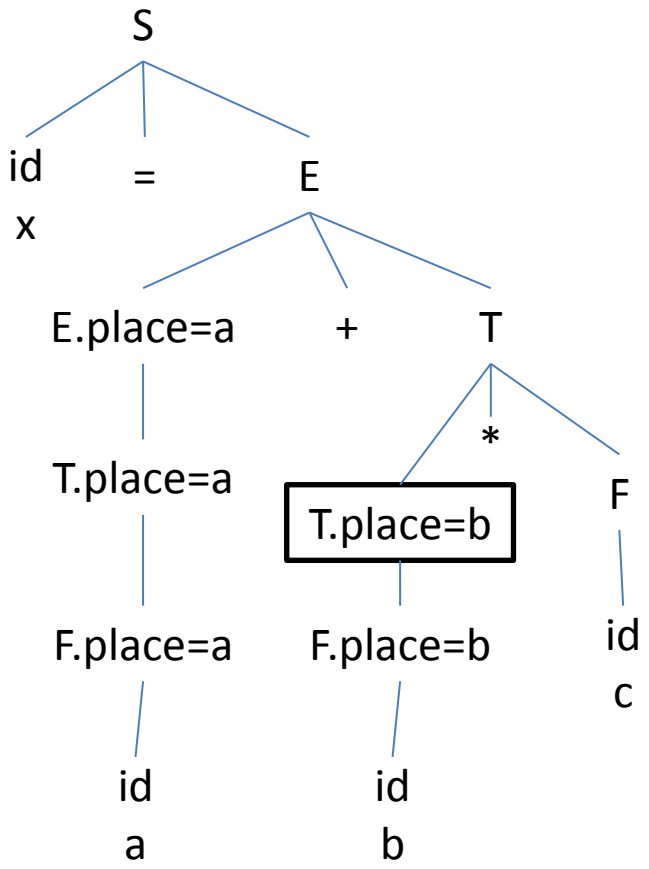
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

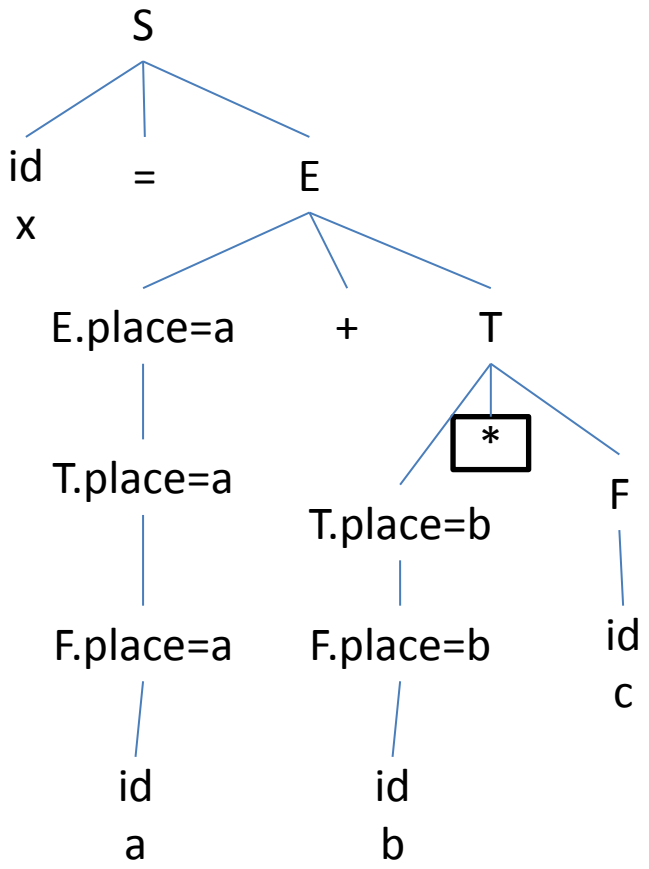
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

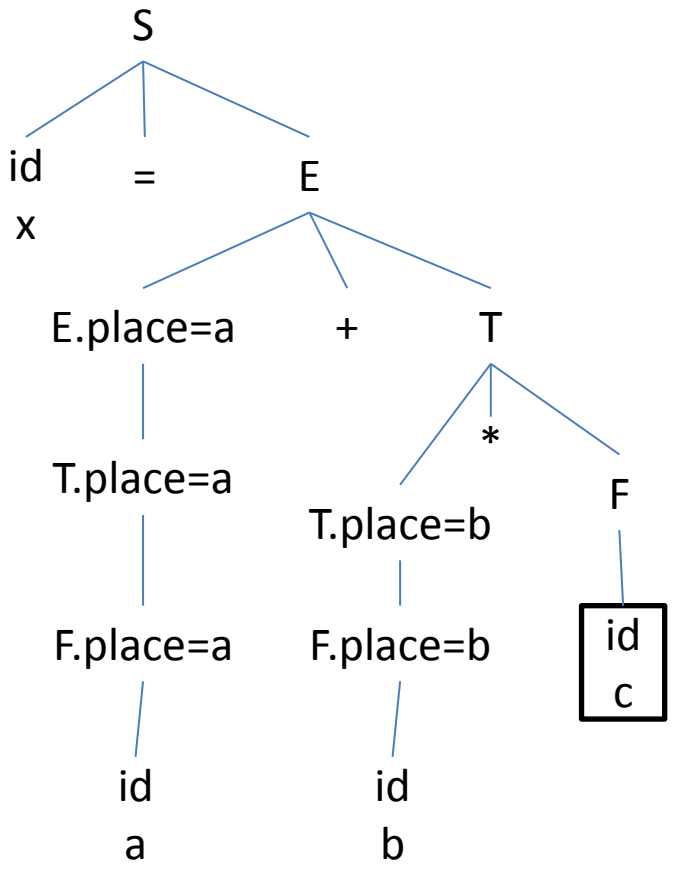
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

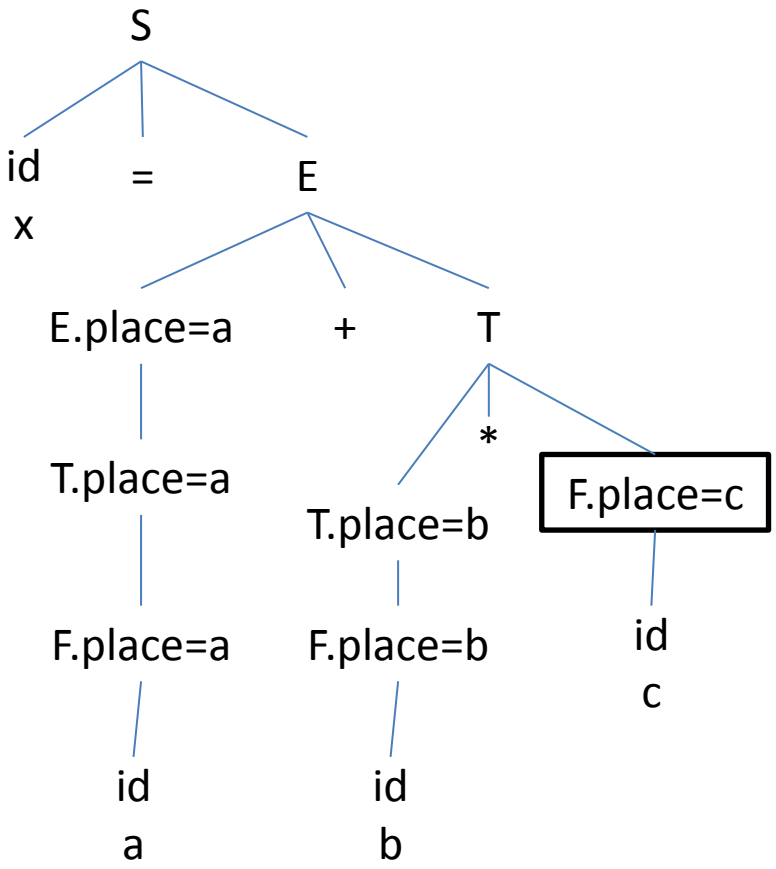
E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

E → T {E.place = T.place}

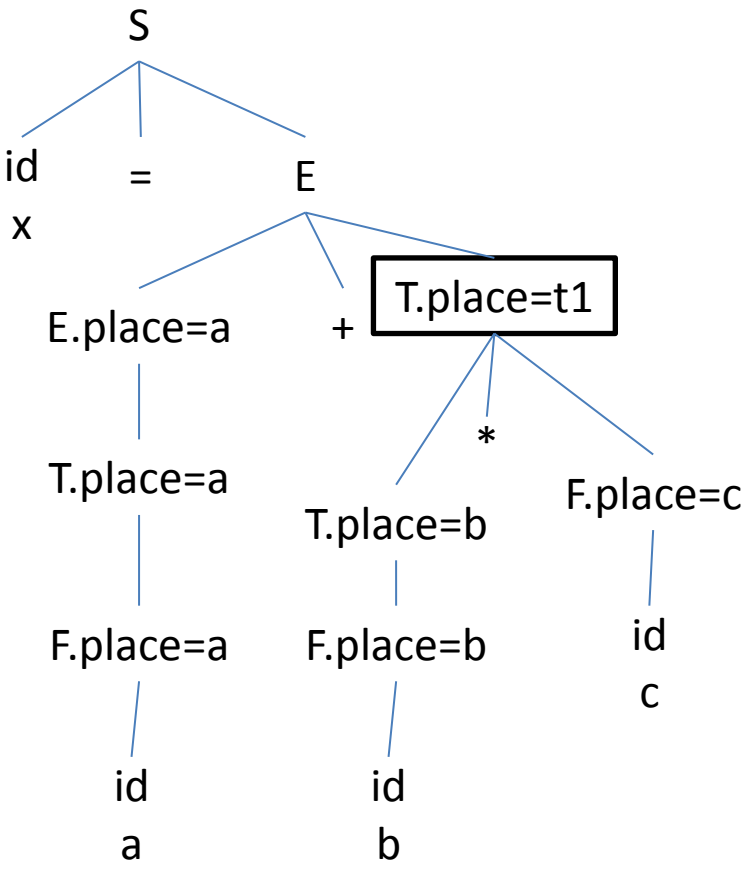
T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c

t1 = b * c



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

E → T {E.place = T.place}

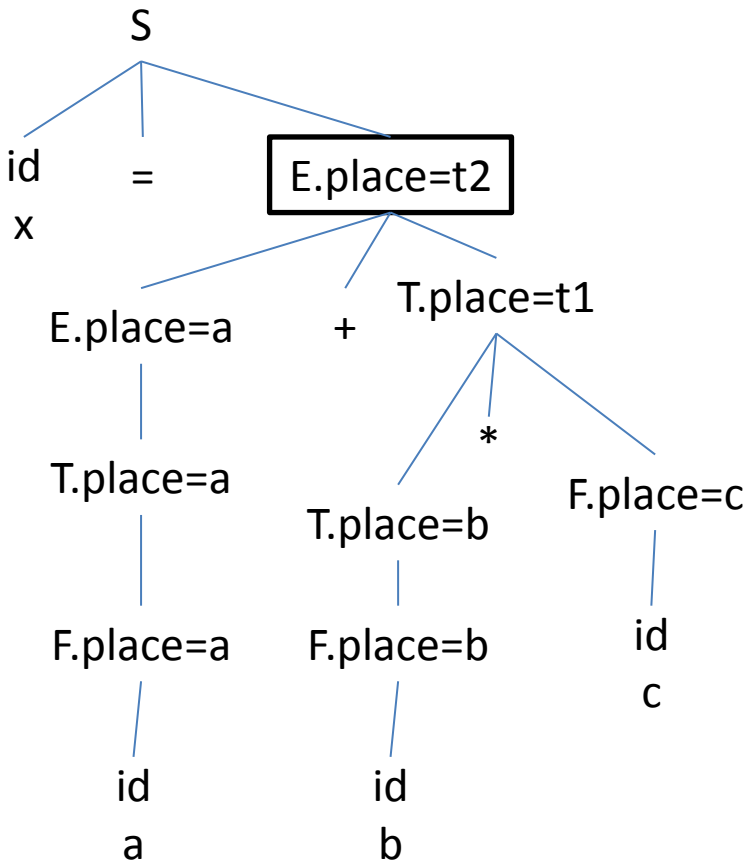
T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}

x = a + b * c

```
t1 = b * c
t2 = a + t1
```



Example 6

- SDT to generate three address code

S → id = E {gen(id.name = E.place}

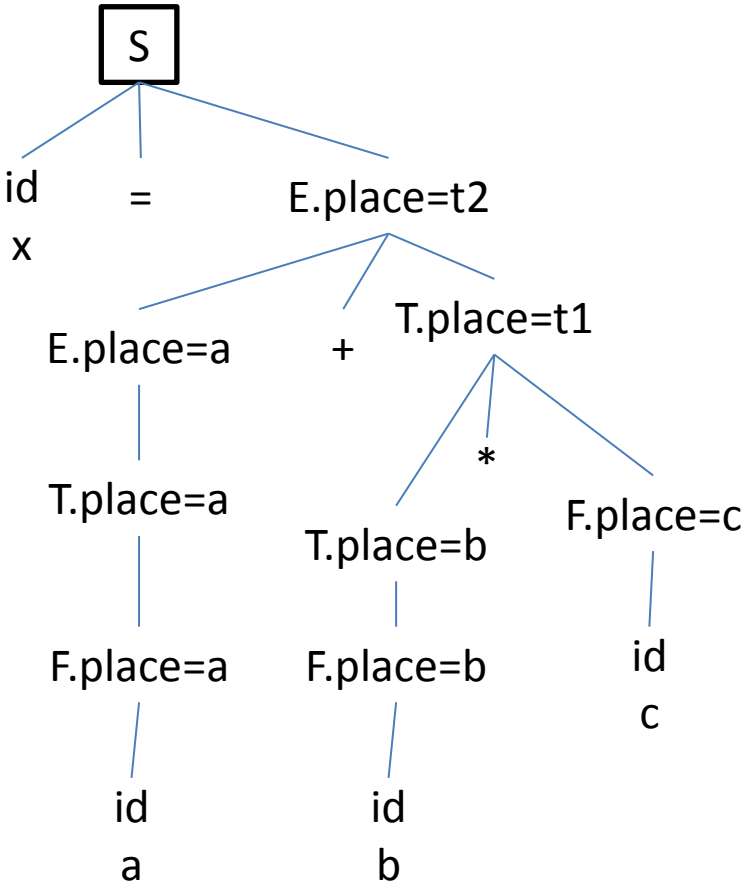
E → E_1 + T {E.place = newTemp();
gen(E.place = E_1.place + T.place);}

E → T {E.place = T.place}

T → T_1 * F {T.place = newTemp();
gen(E.place = T_1.place * F.place);}

T → F {T.place = F.place}

F → id {F.place = id.name}



x = a + b * c

```
t1 = b * c
t2 = a + t1
x = t2
```