Example 7

- SDT to add type information in symbol table

- int x, y, z

x	int
y	int
z	int

Example 7

- SDT to add type information in symbol table

D → TL	{L.in=T.type}
T → int	{T.type = int}
T → char	{T.type=char}
L → L_1 ,id	{L_1.in = L.in; addtype(id.name , L_1.in)}
L → id	{addtype(id.name , L.in)}

The in attribute of L is an inherited attribute that takes the value of type attribute of T.

addtype() :- adds the type information in symbol table

Example 7

- SDT to add type information in symbol table

$D \rightarrow TL$	{L.in=T.type}	inherited attribute
$T \rightarrow int$	{T.type = int}	synthesized attribute
$T \rightarrow char$	{T.type=char}	synthesized attribute
$L \rightarrow L_1$,id	{L_1.in = L.in; addtype(id.name , L_1.in)}	inherited attribute
$L \rightarrow id$	{addtype(id.name , L.in)}	

addtype() :- adds the type information in symbol table

Example 7

- SDT to add type information in symbol table

L-attributed definition

$D \rightarrow TL$	{L.in=T.type}	inherited attribute
$T \rightarrow int$	{T.type = int}	synthesized attribute
$T \rightarrow char$	{T.type=char}	synthesized attribute
$L \rightarrow L_1$,id	{L_1.in = L.in; addtype(id.name , L_1.in)}	inherited attribute
$L \rightarrow id$	{addtype(id.name , L.in)}	

addtype() :- adds the type information in symbol table

Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

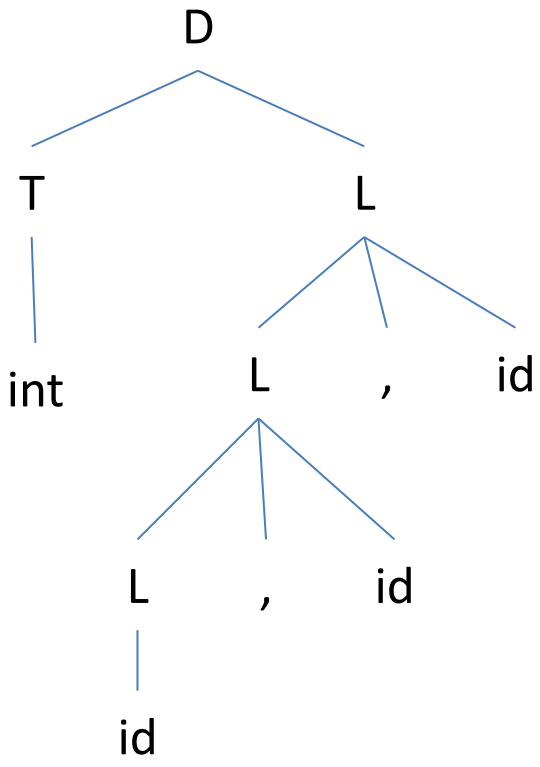
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

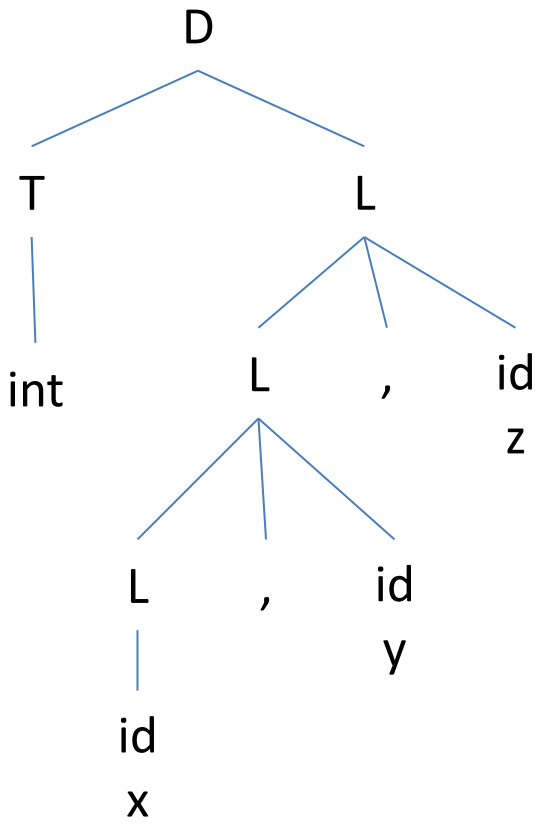
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

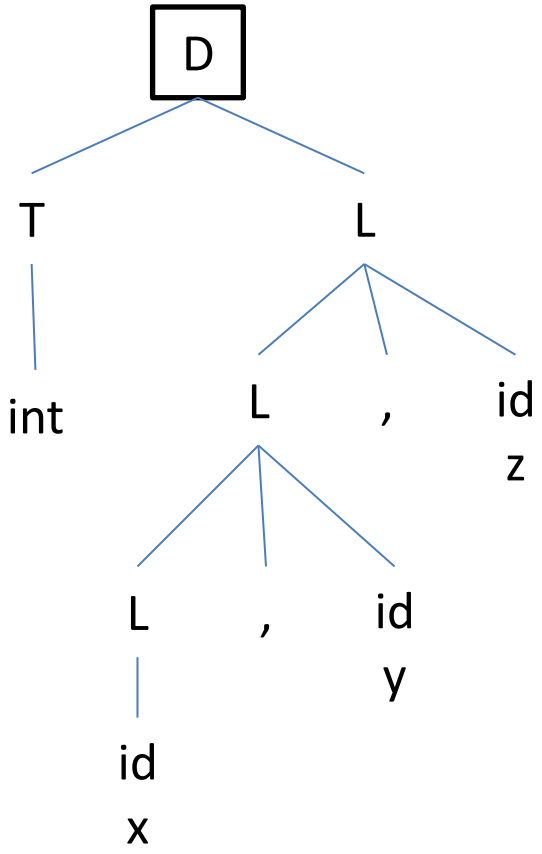
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

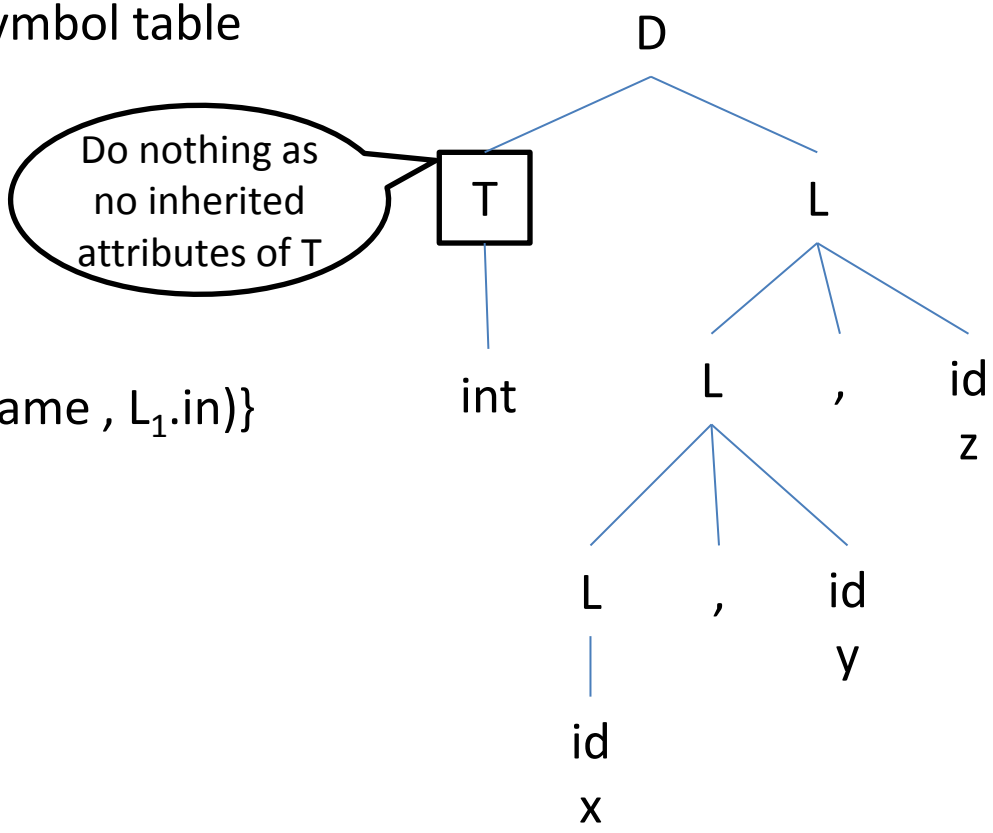
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

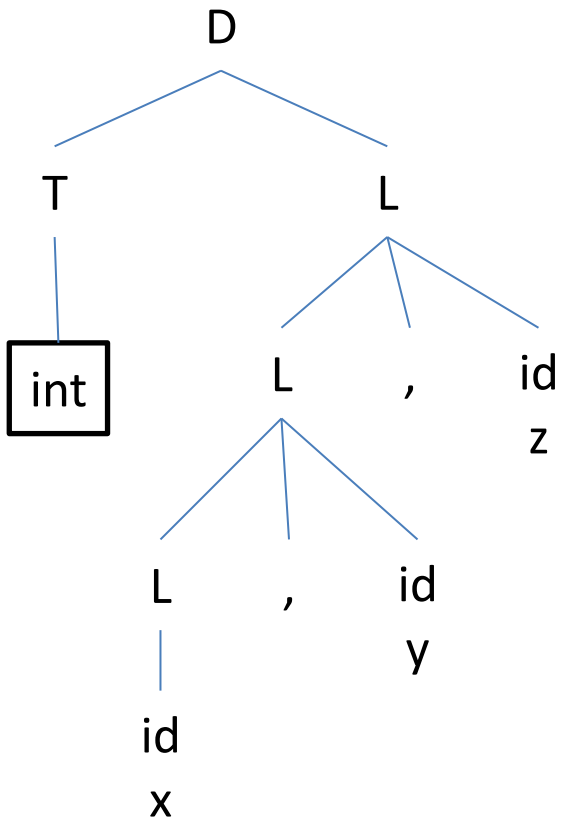
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

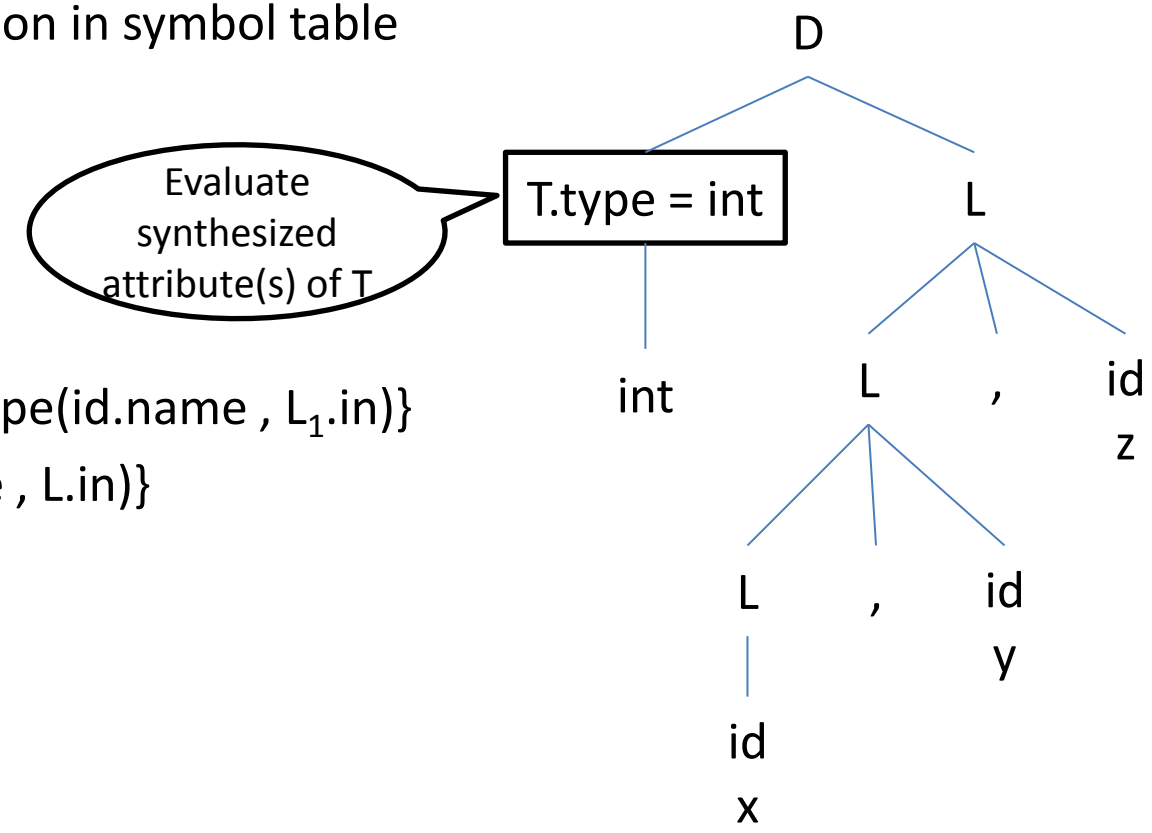
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

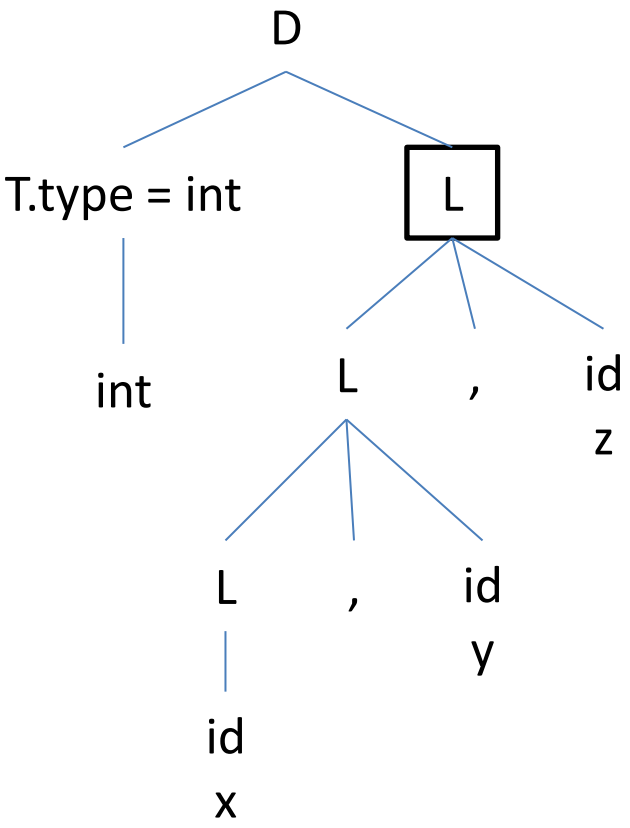
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

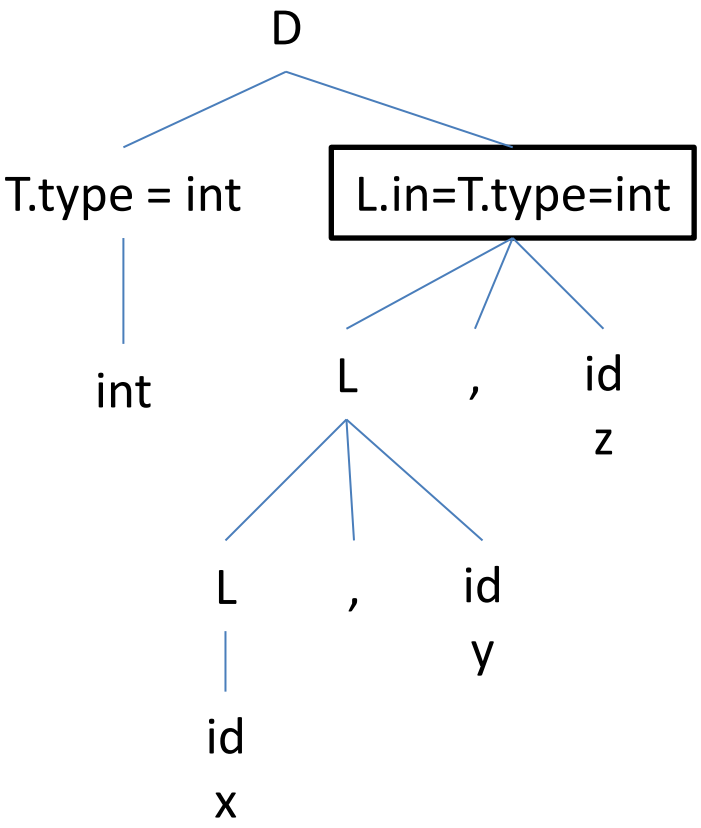


Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL	{L.in = T.type}
T → int	{T.type = int}
T →char	{T.type = char}
L → L_1 ,id	{L_1.in = L.in; addtype(id.name , L_1.in)}
L→ id	{addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

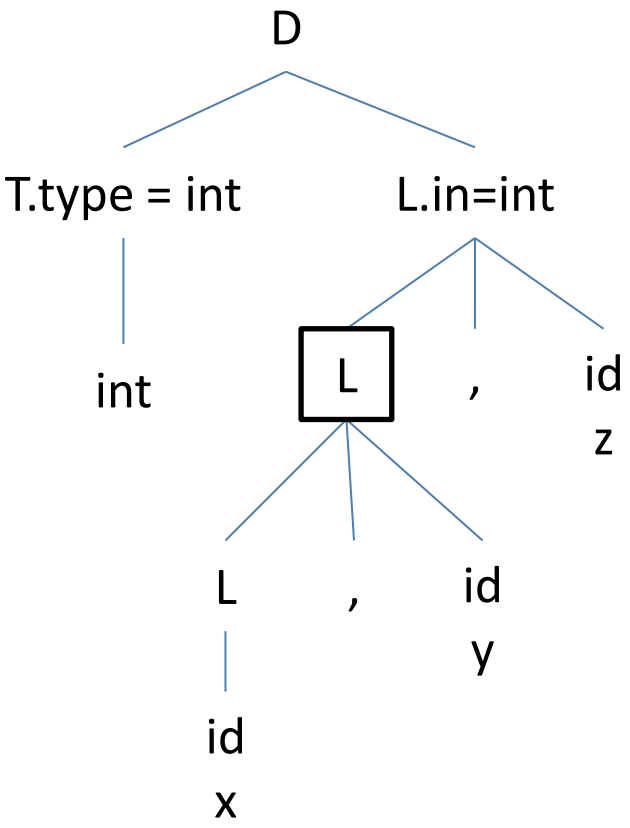
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

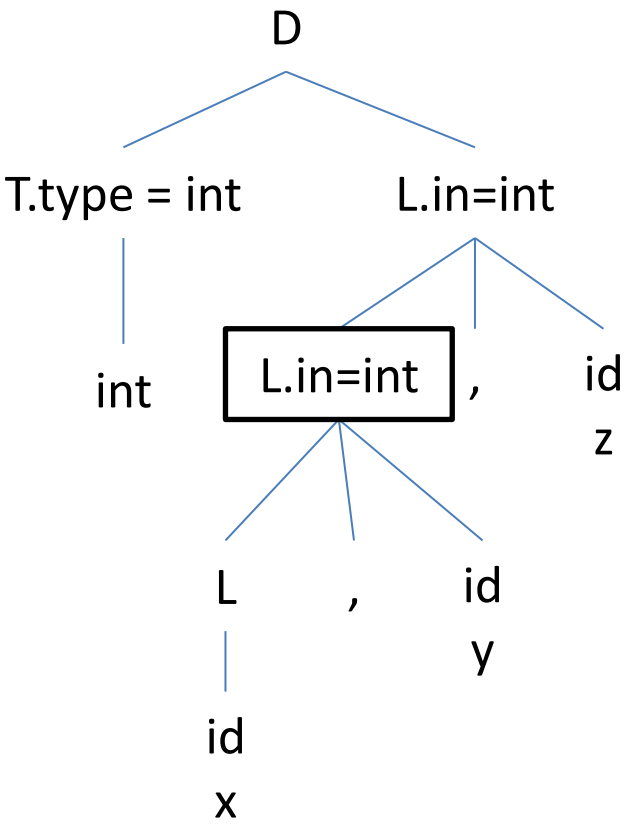
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

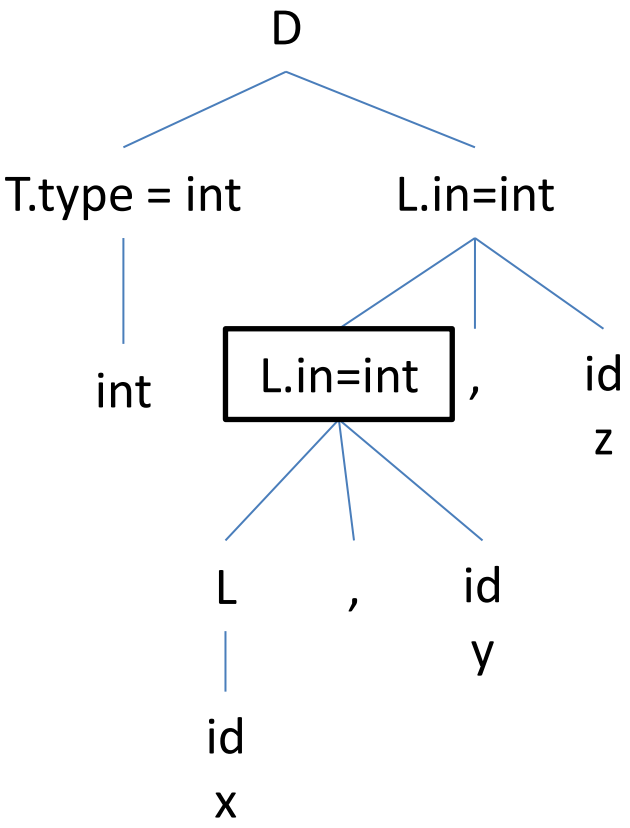
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

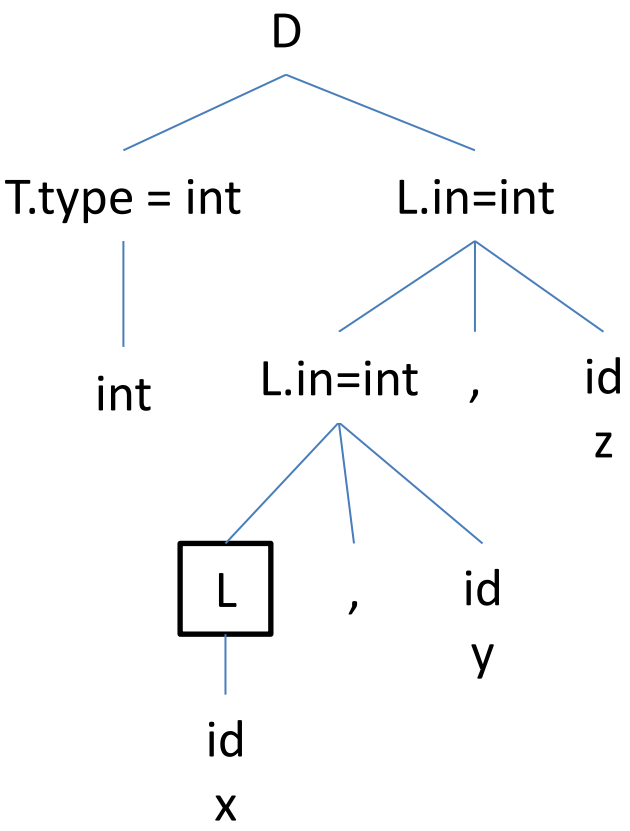
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

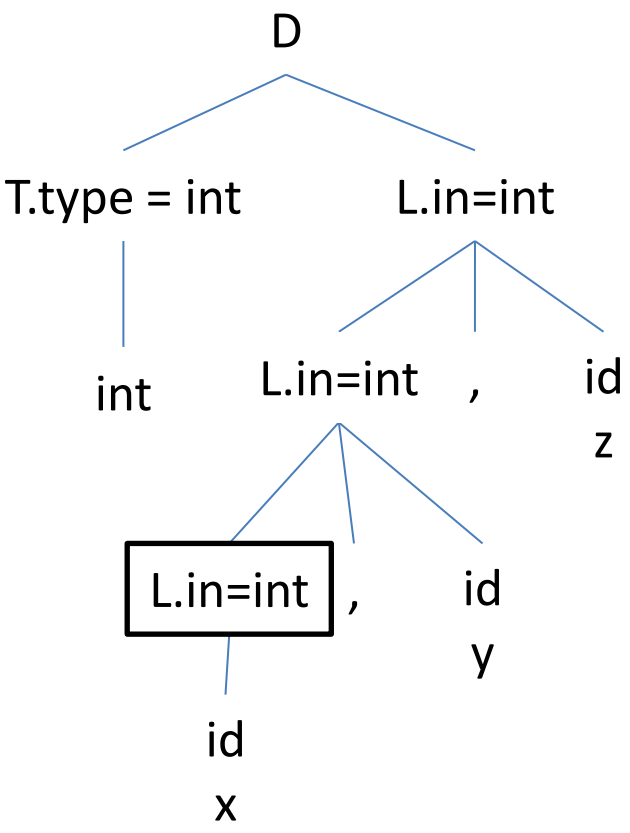
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

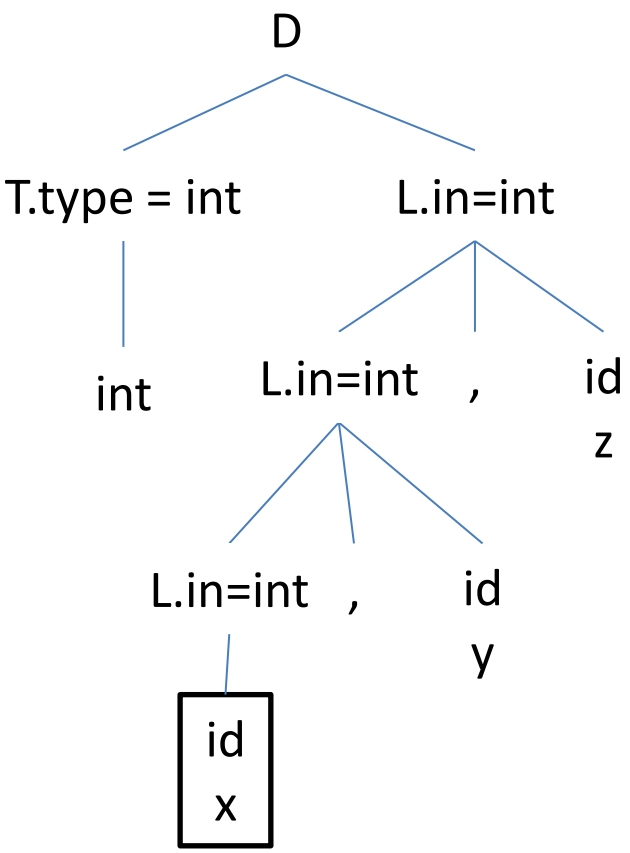
T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

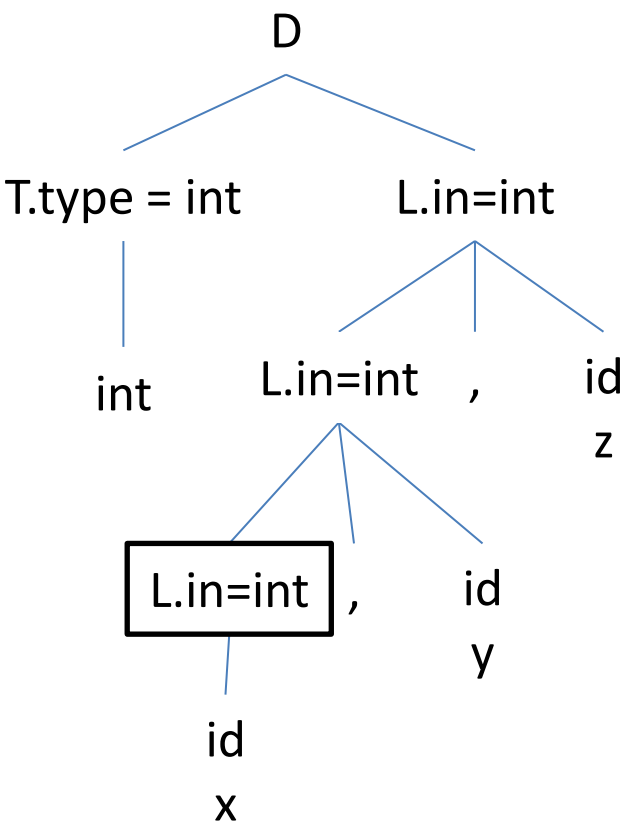
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

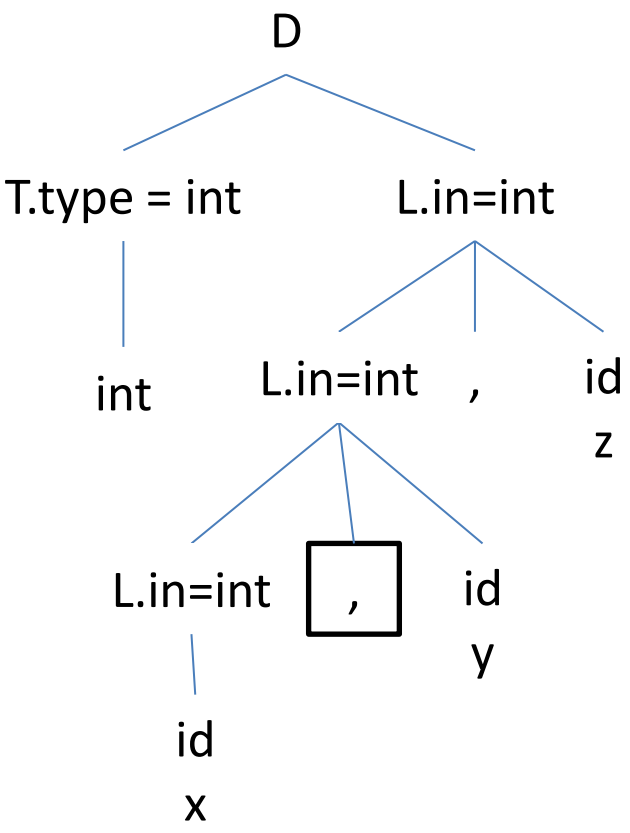
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

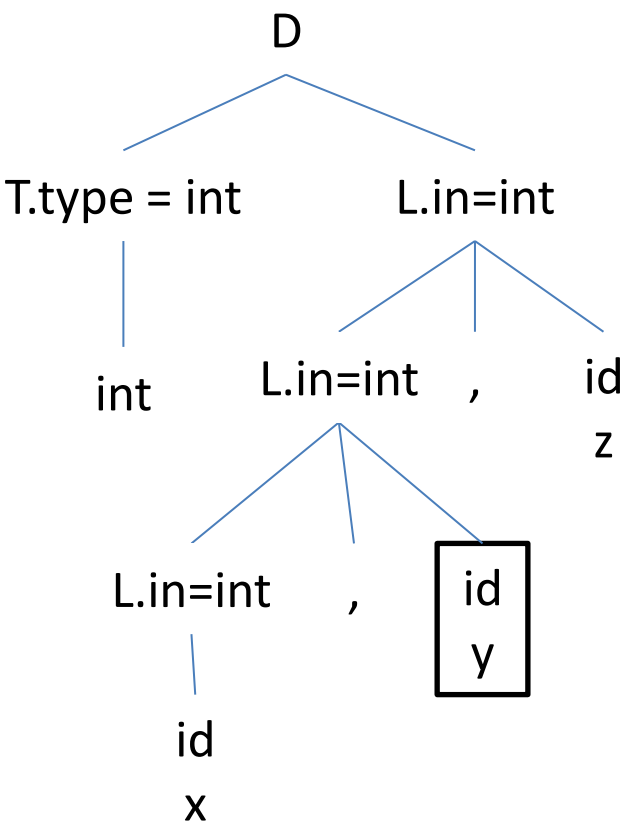
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

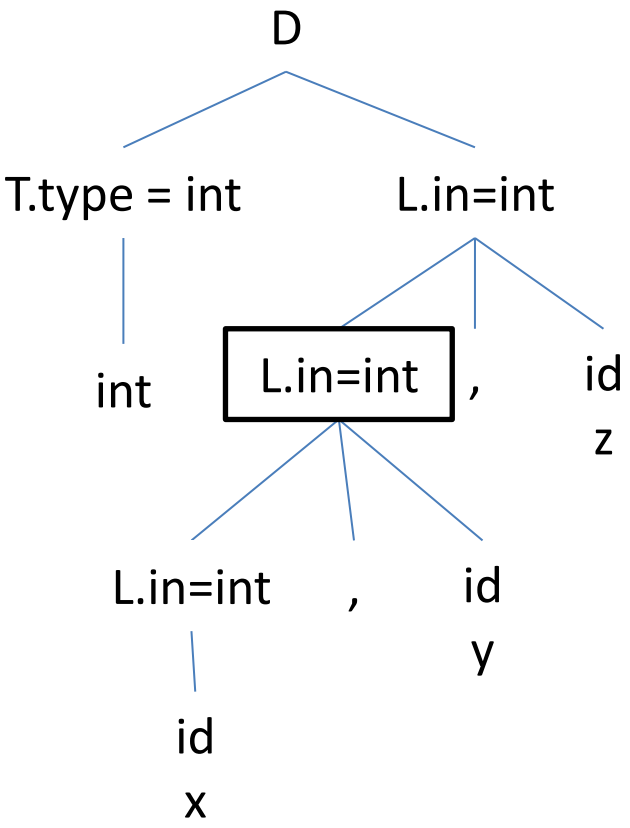
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int
y	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

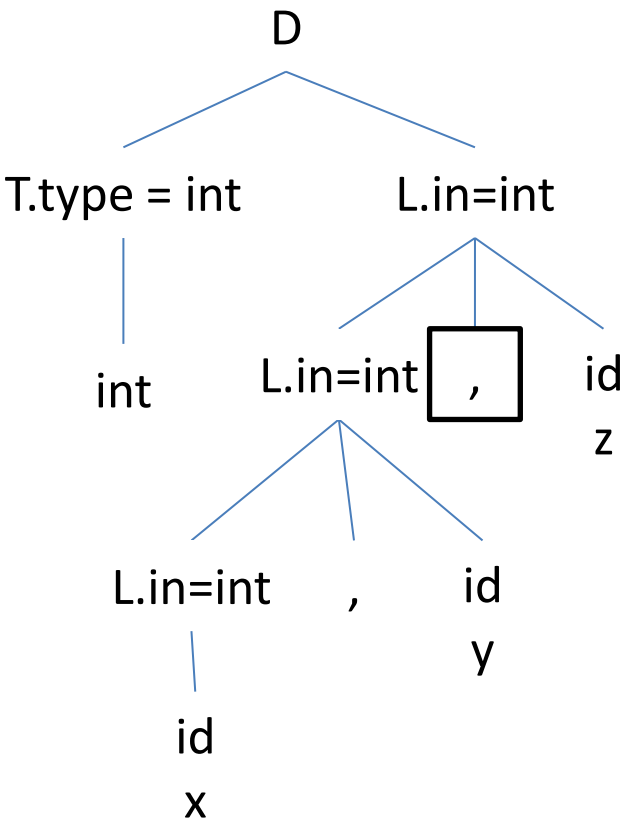
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int
y	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

D → TL {L.in = T.type}

T → int {T.type = int}

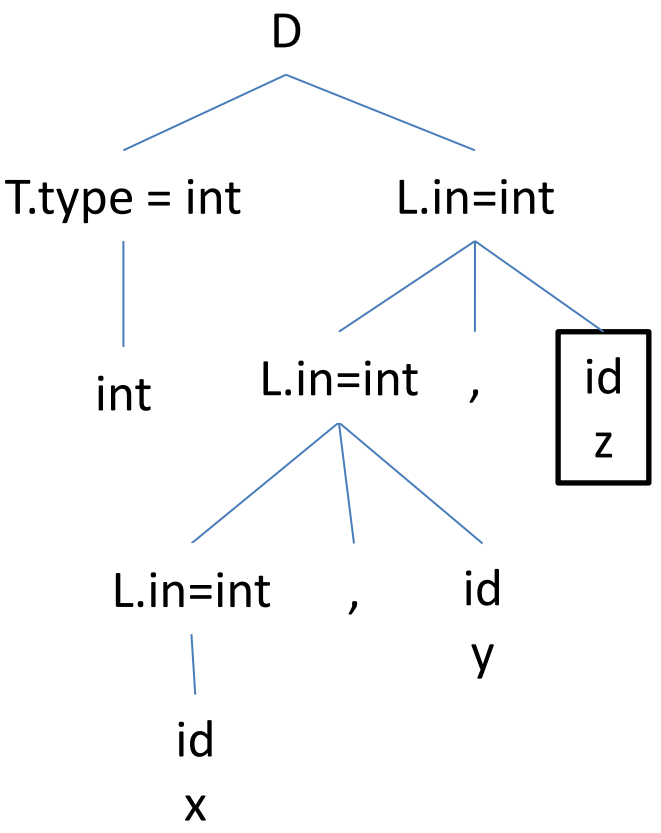
T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}

int x, y, z

x	int
y	int



Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

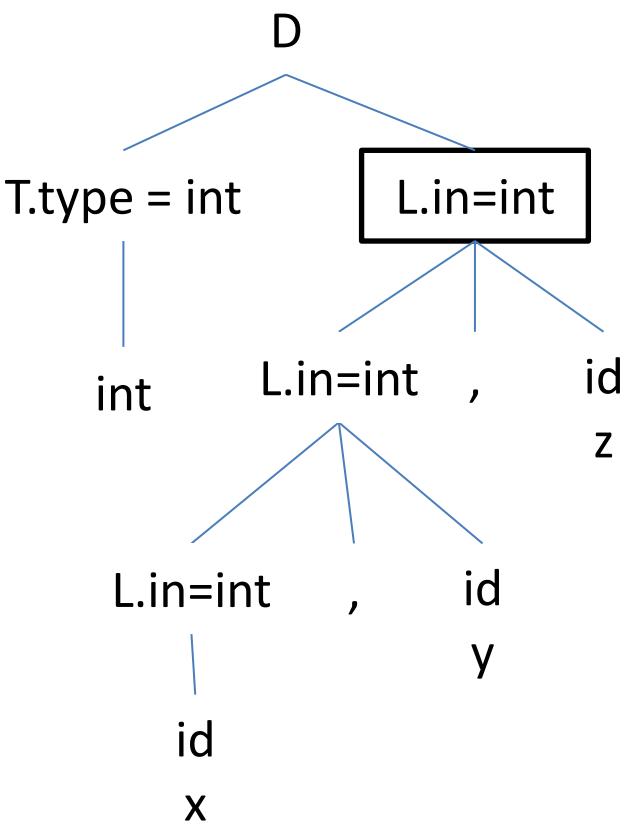
D → TL {L.in = T.type}

T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}



int x, y, z

x	int
y	int
z	int

Example 7
[L-attributed:- top-down left to right]

- SDT to add type information in symbol table

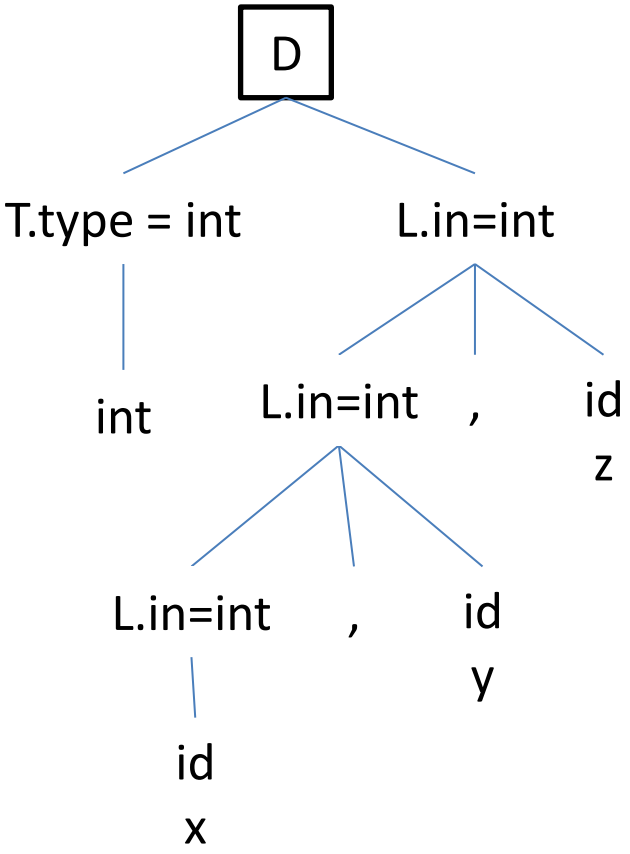
D → TL {L.in = T.type}

T → int {T.type = int}

T →char {T.type = char}

L → L_1 ,id {L_1.in = L.in; addtype(id.name , L_1.in)}

L→ id {addtype(id.name , L.in)}



int x, y, z

x	int
y	int
z	int

Example 7 (second method)

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type), D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

add_type() :- adds the type information in symbol table

Example 7 (second method)

- SDT to add type information in symbol table

S-attributed definition

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type), D.type=D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

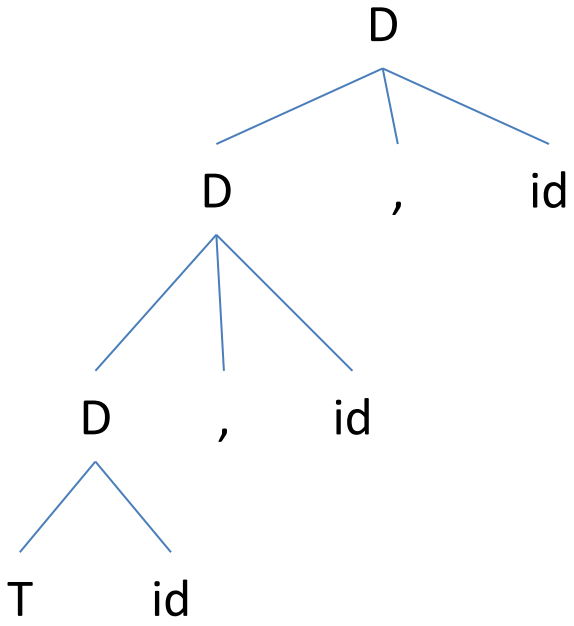
add_type() :- adds the type information in symbol table

Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

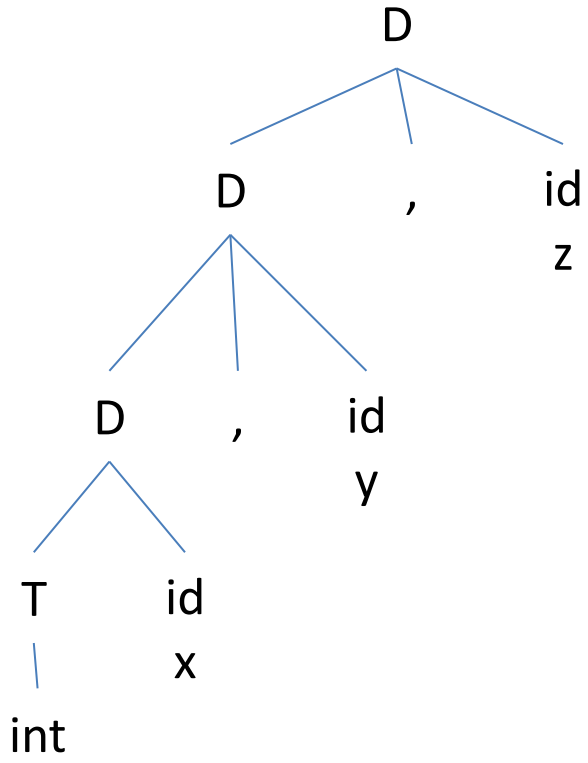


Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

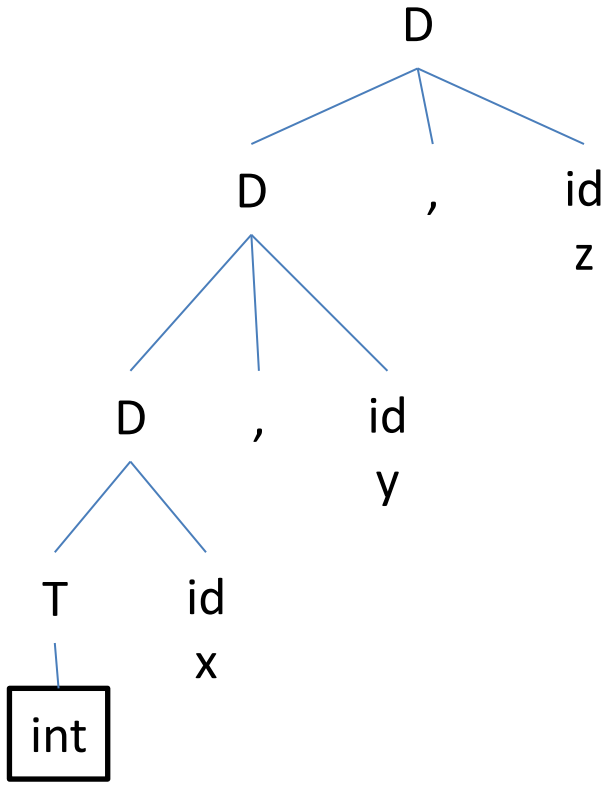


Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

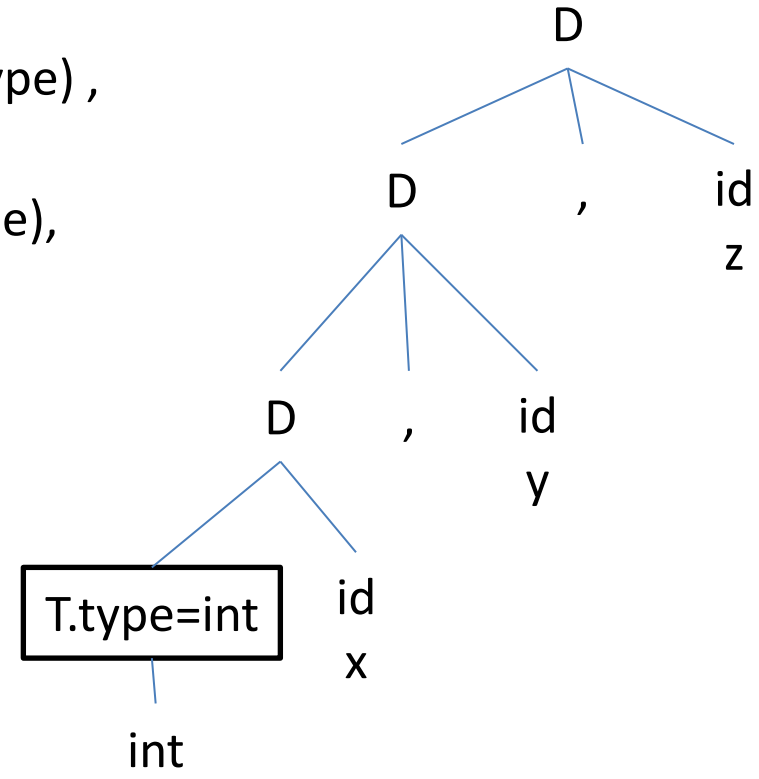


Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

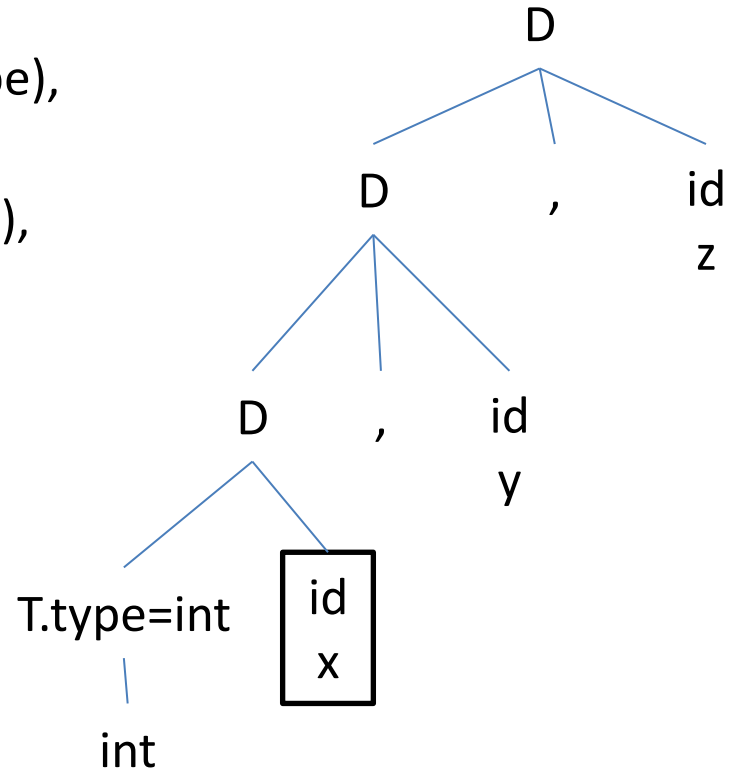


Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type), D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z



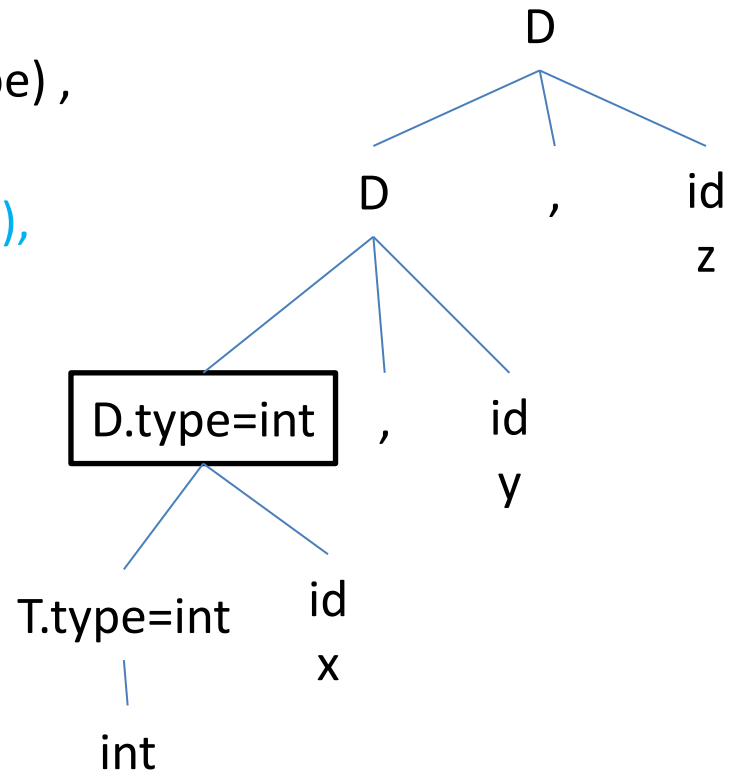
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int



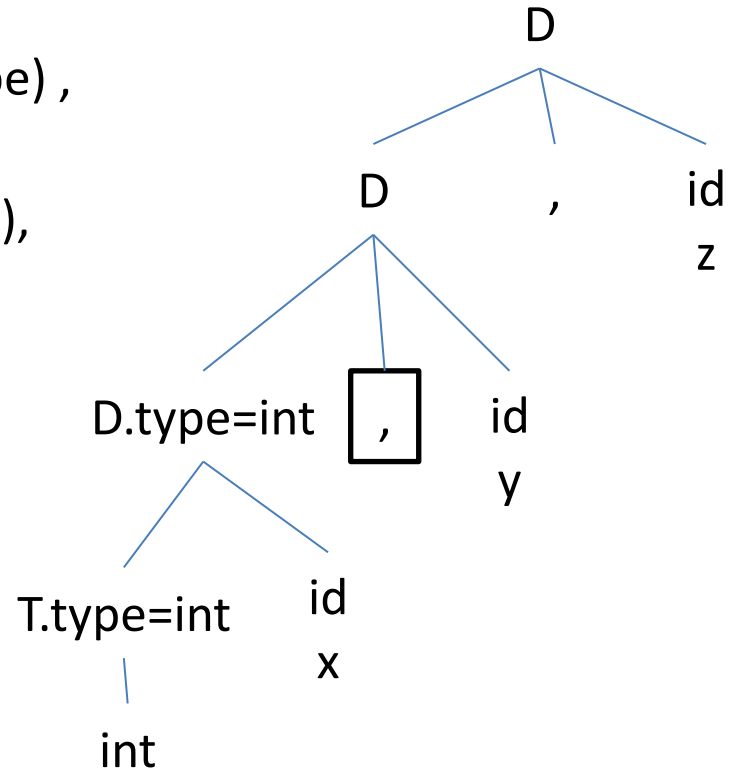
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int



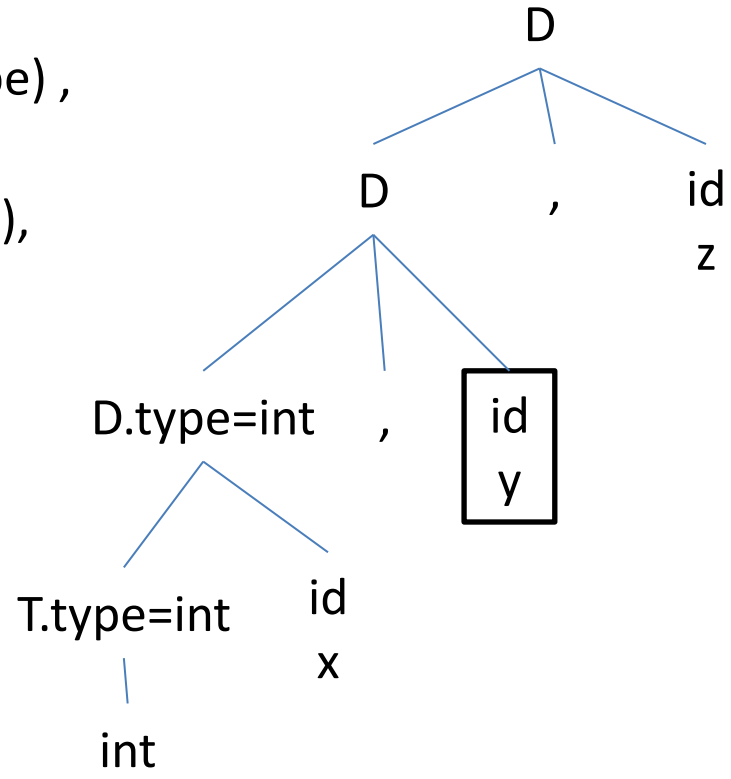
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int



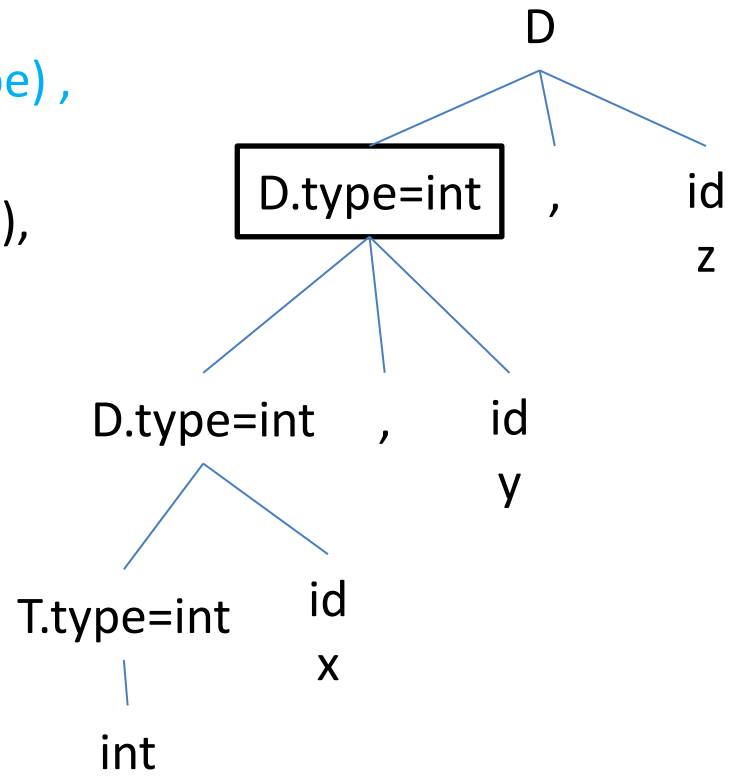
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int
y	int



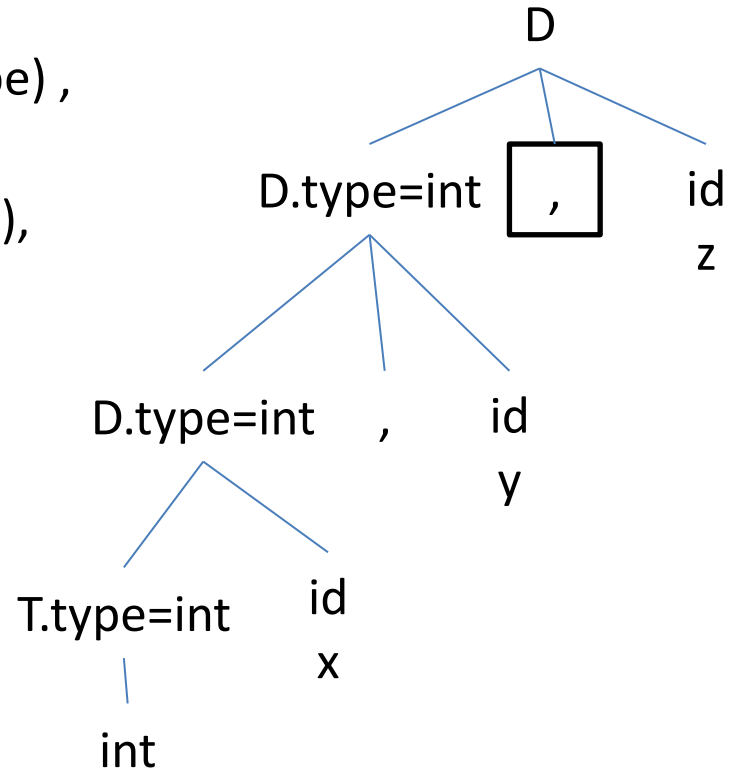
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int
y	int



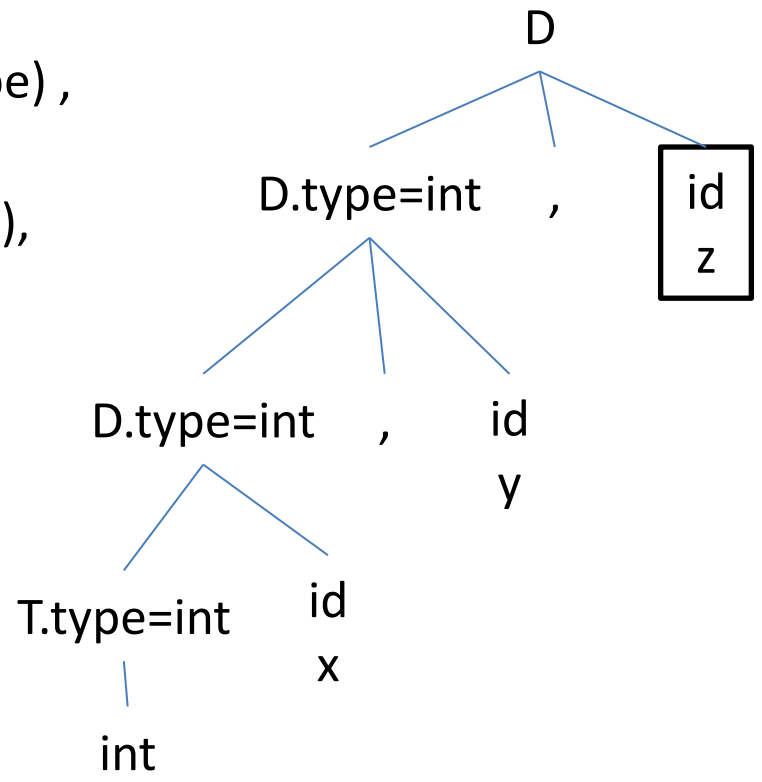
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int
y	int



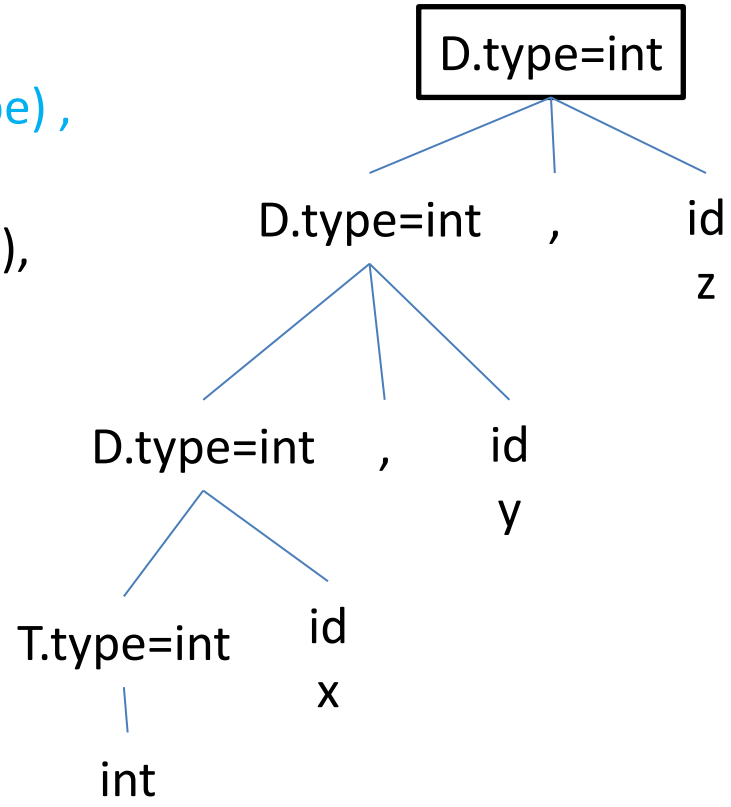
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int
y	int
z	int



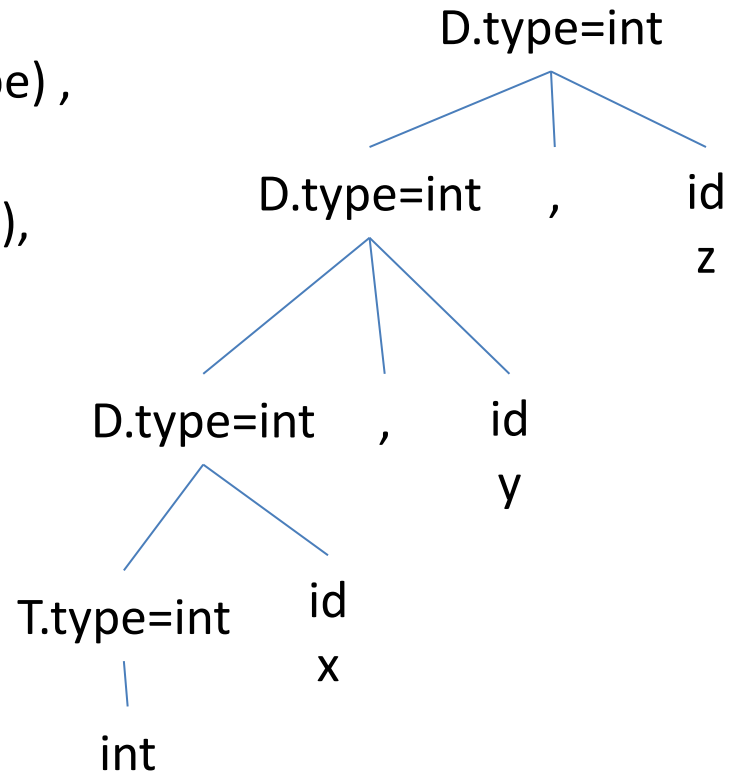
Example 7 (second method) [S-attributed:- bottom-up left to right]

- SDT to add type information in symbol table

$D \rightarrow D_1$, id	{add_type(id.name, D_1.type) , D.type= D_1.type}
D → T id	{add_type(id.name, T.type), D.type=T.type}
T → int	{T.type = int}
T → char	{T.type=char}

int x, y, z

x	int
y	int
z	int



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$	{if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}
$E \rightarrow E_1 == E_2$	{if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}
$E \rightarrow (E_1)$	{E.type = E_1.type}
E → num	{E.type = int}
E → true	{E.type = boolean}
E → false	{E.type = boolean}

Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

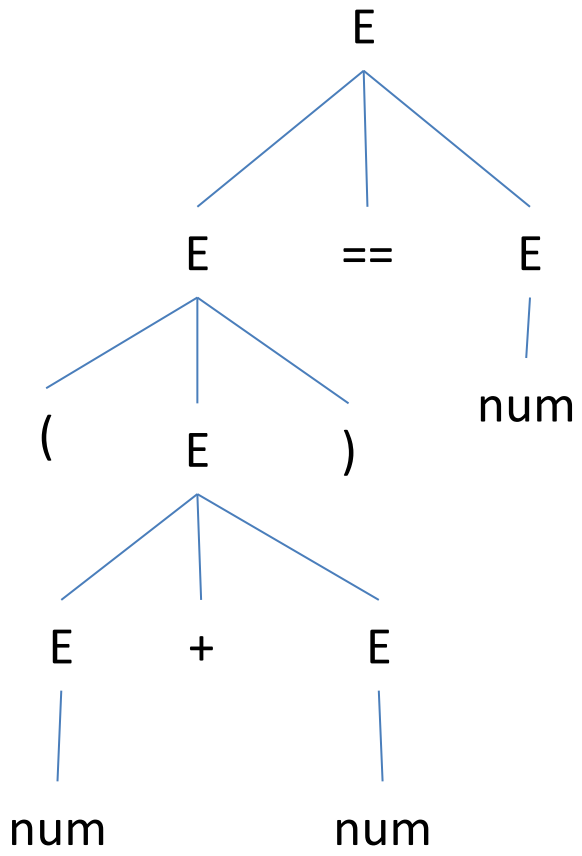
$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}



(2 + 3) == 8

Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

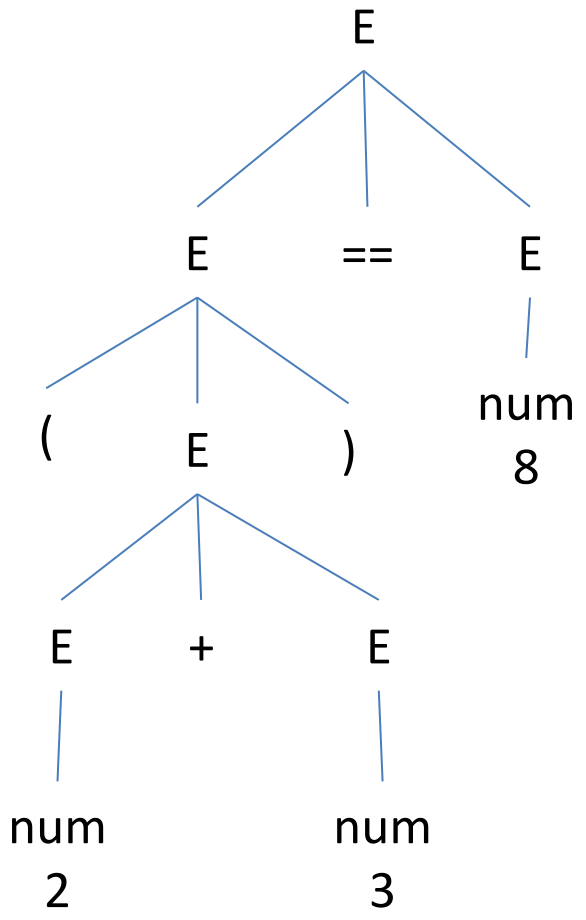
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

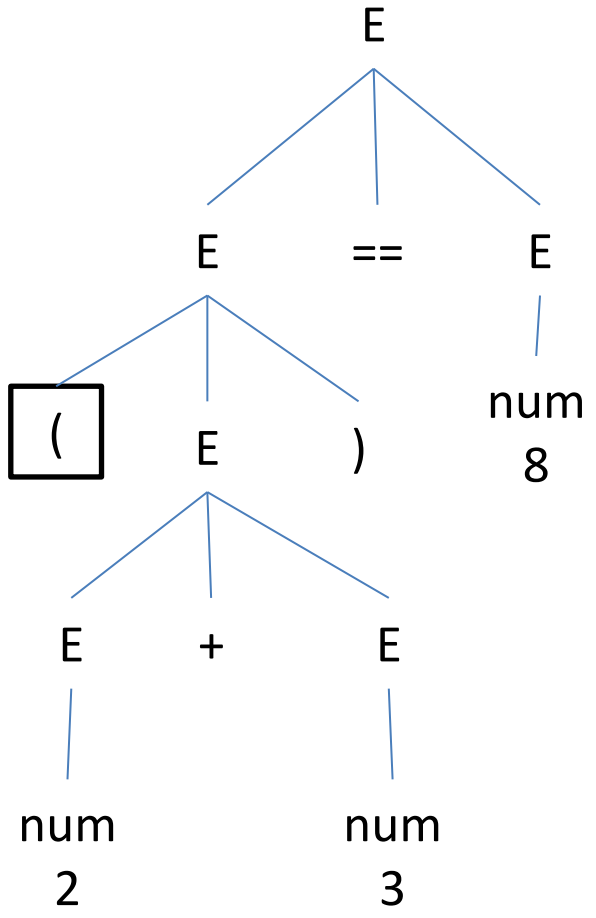
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

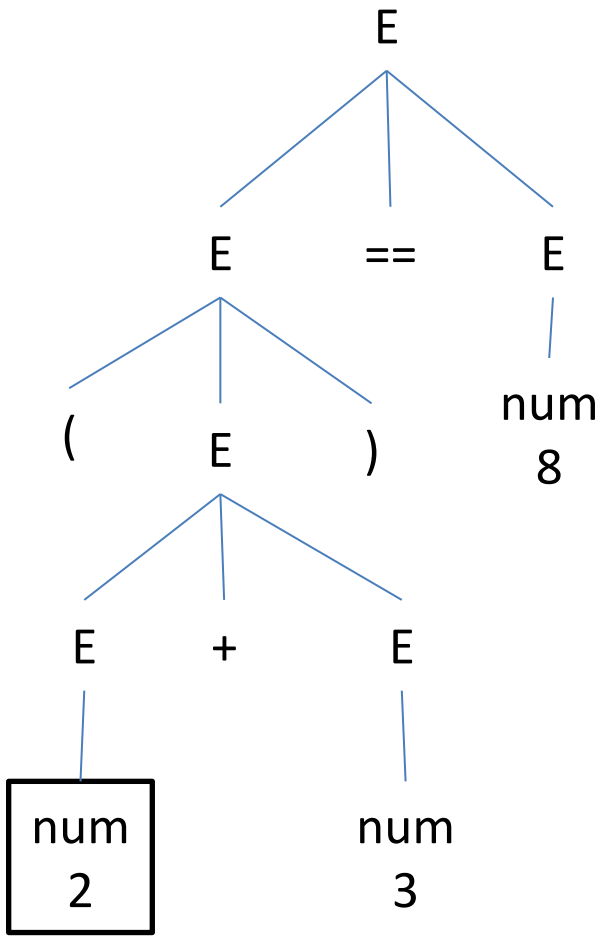
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

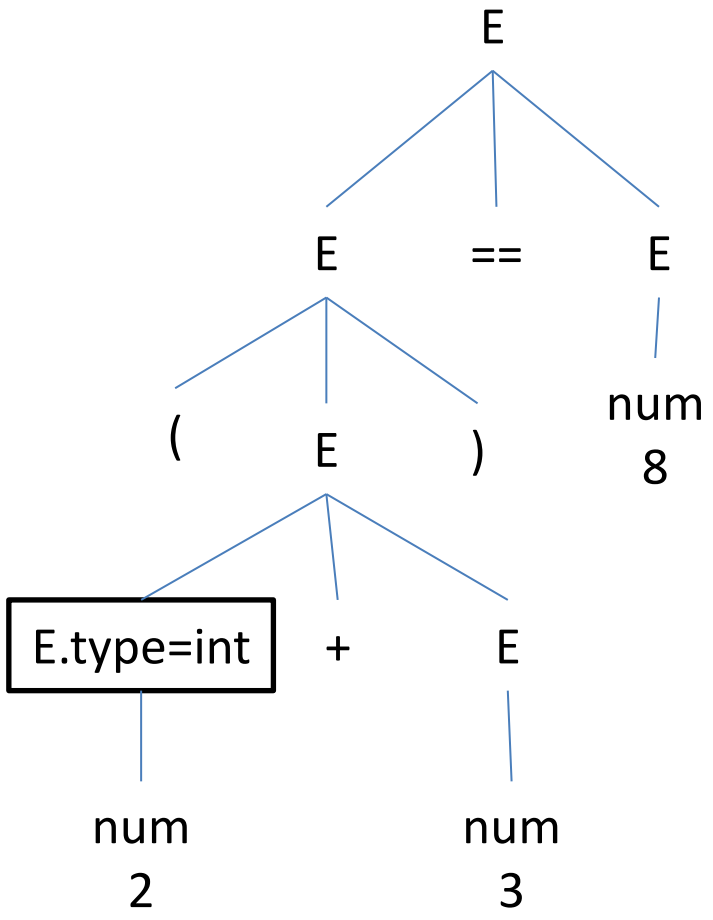
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

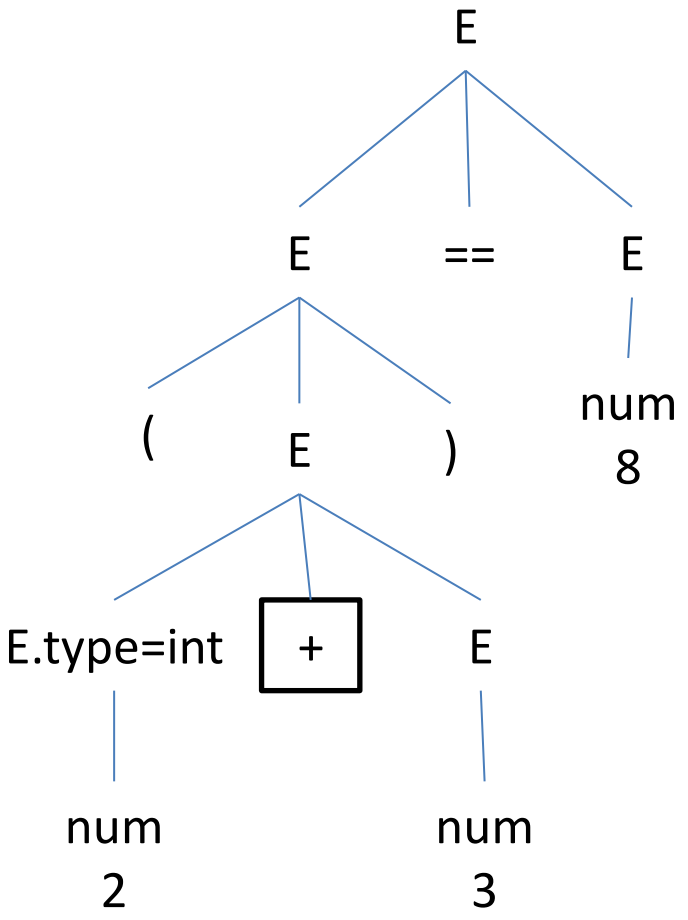
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

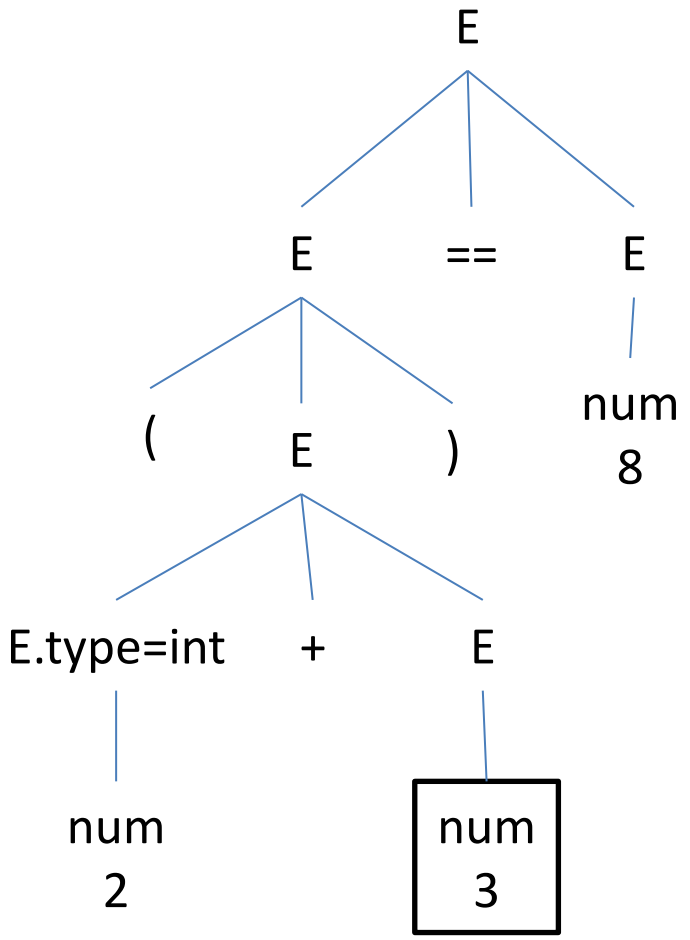
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

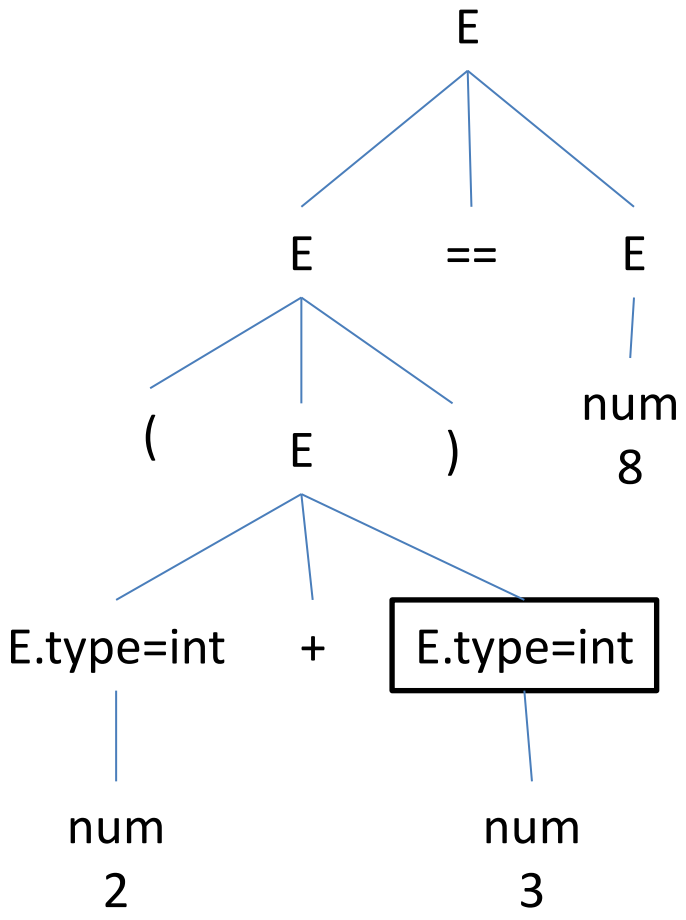
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

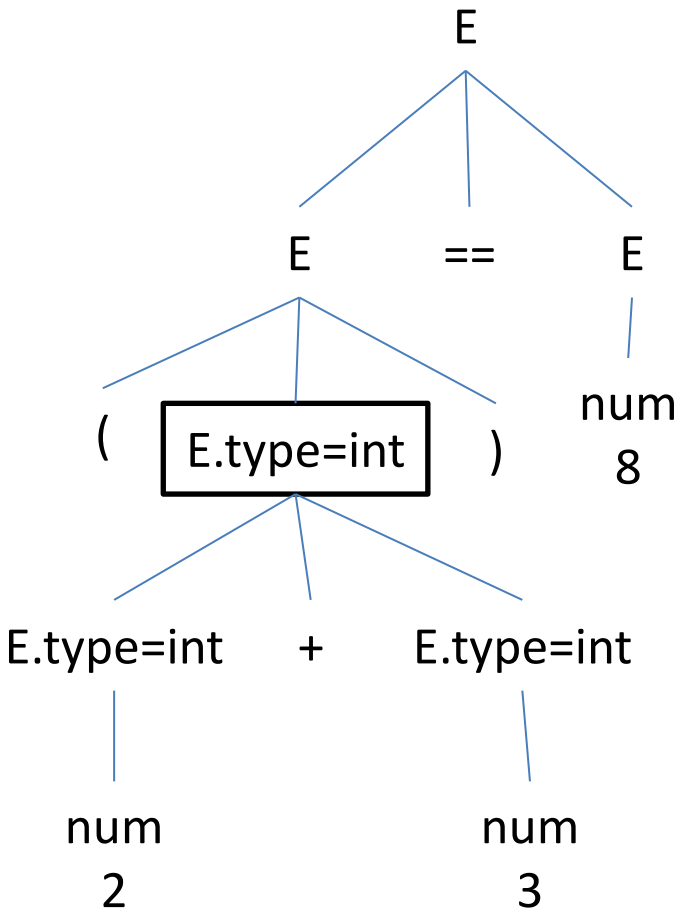
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

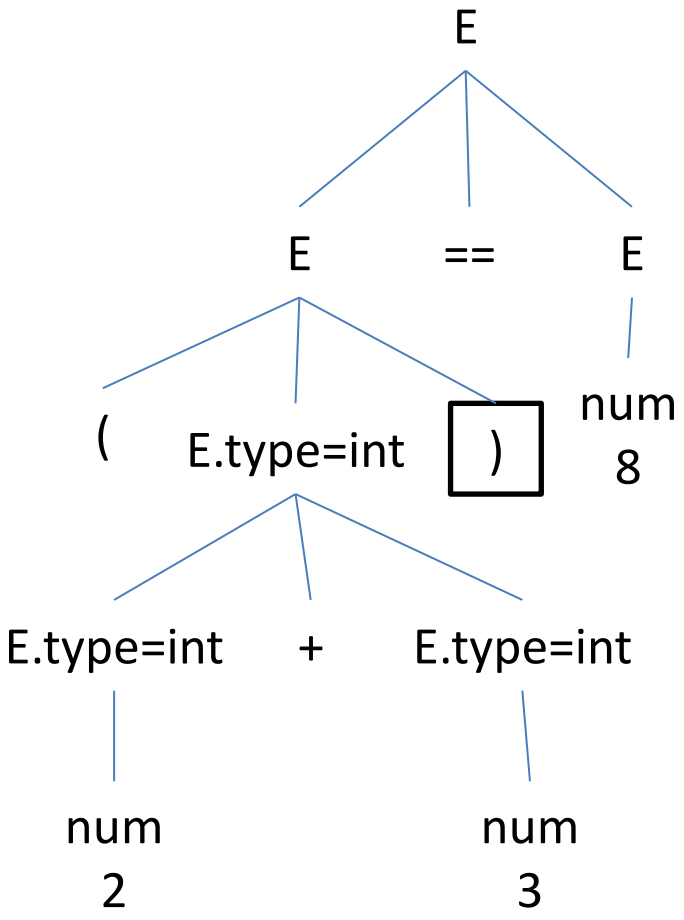
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

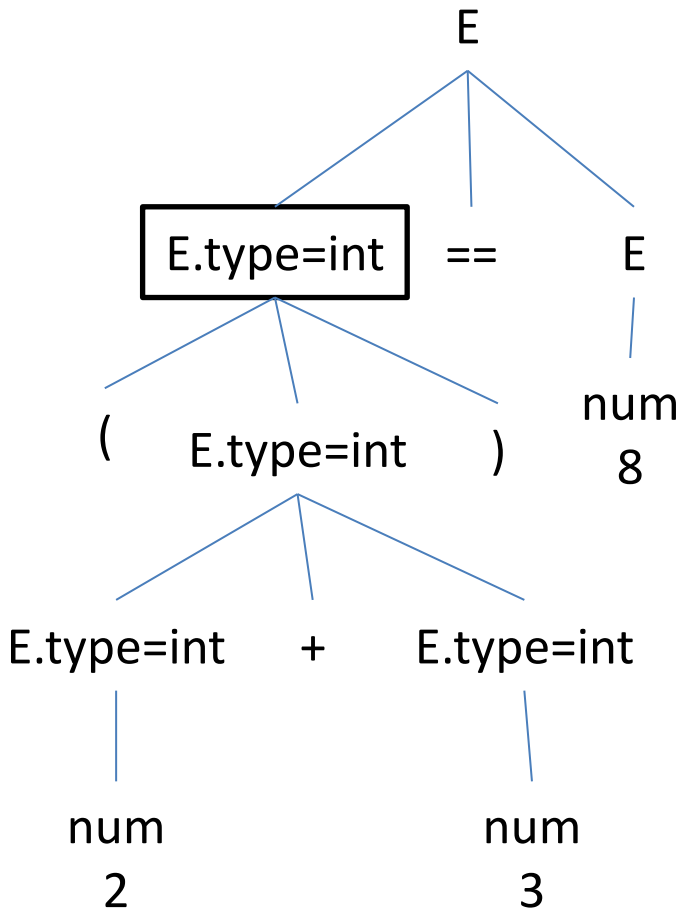
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

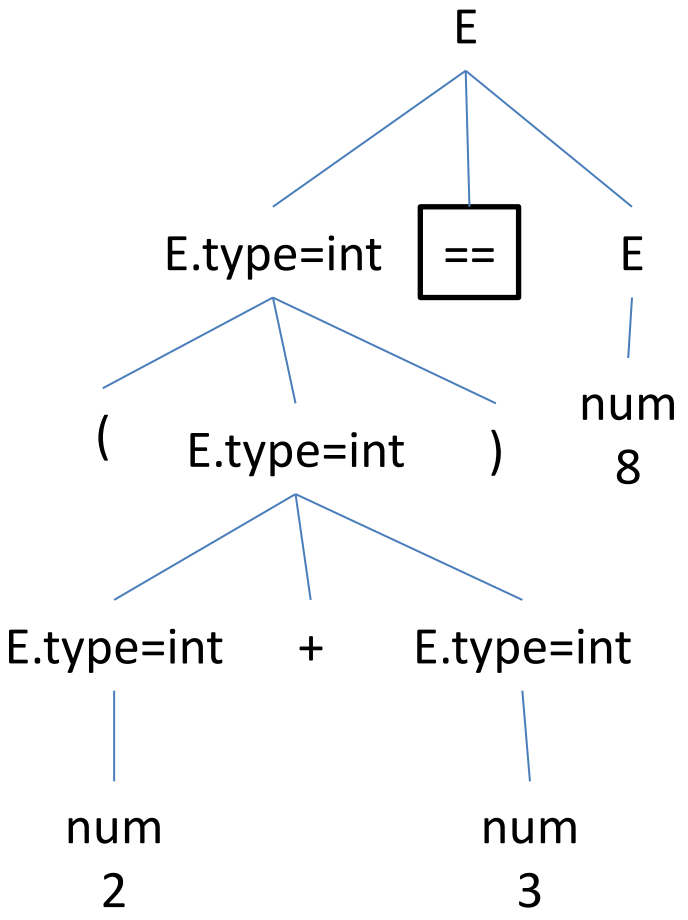
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

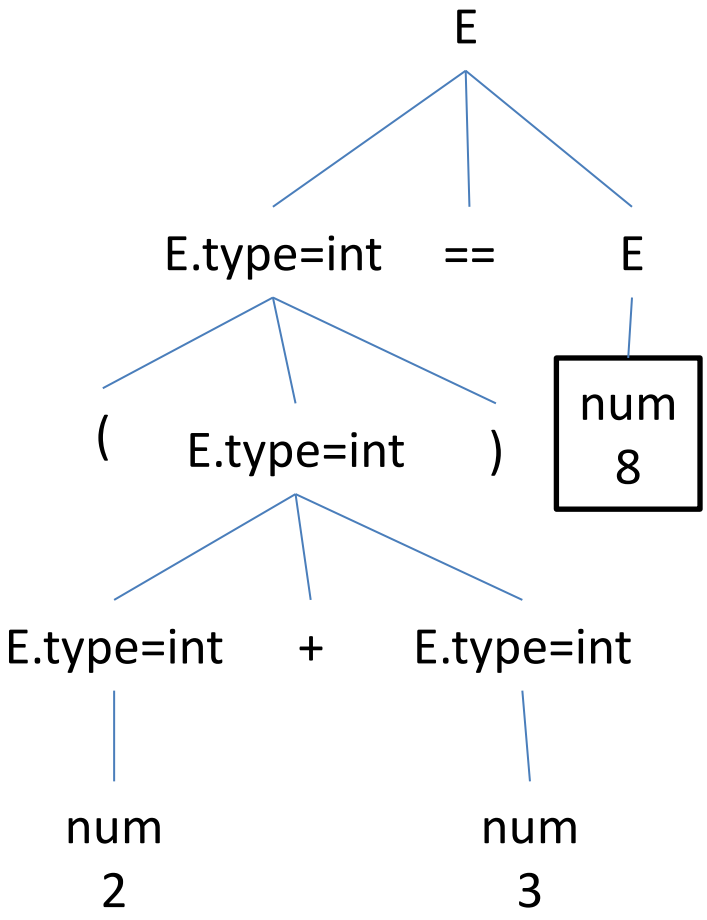
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

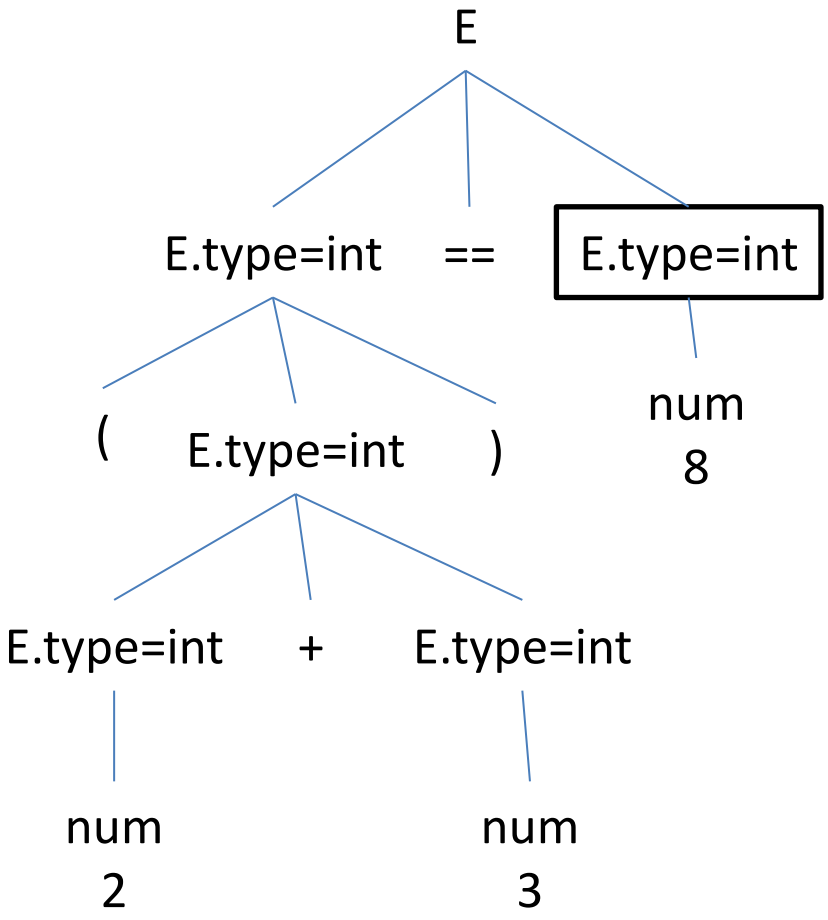
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

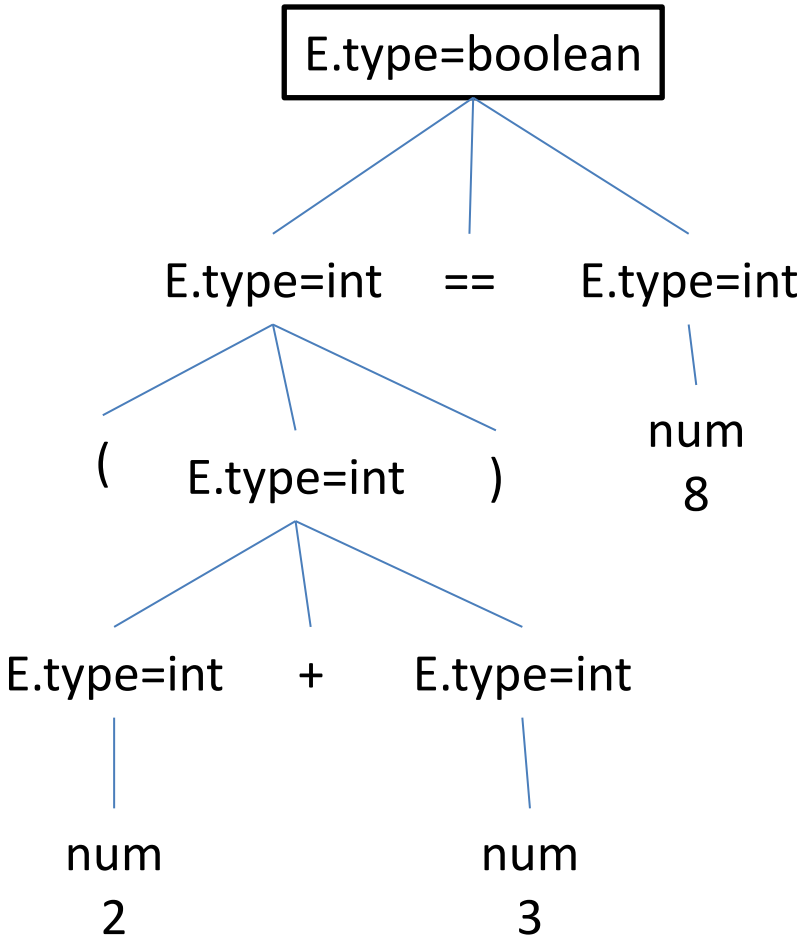
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 + 3) == 8



Example 8
[S-attributed:- bottom-up left to right]

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

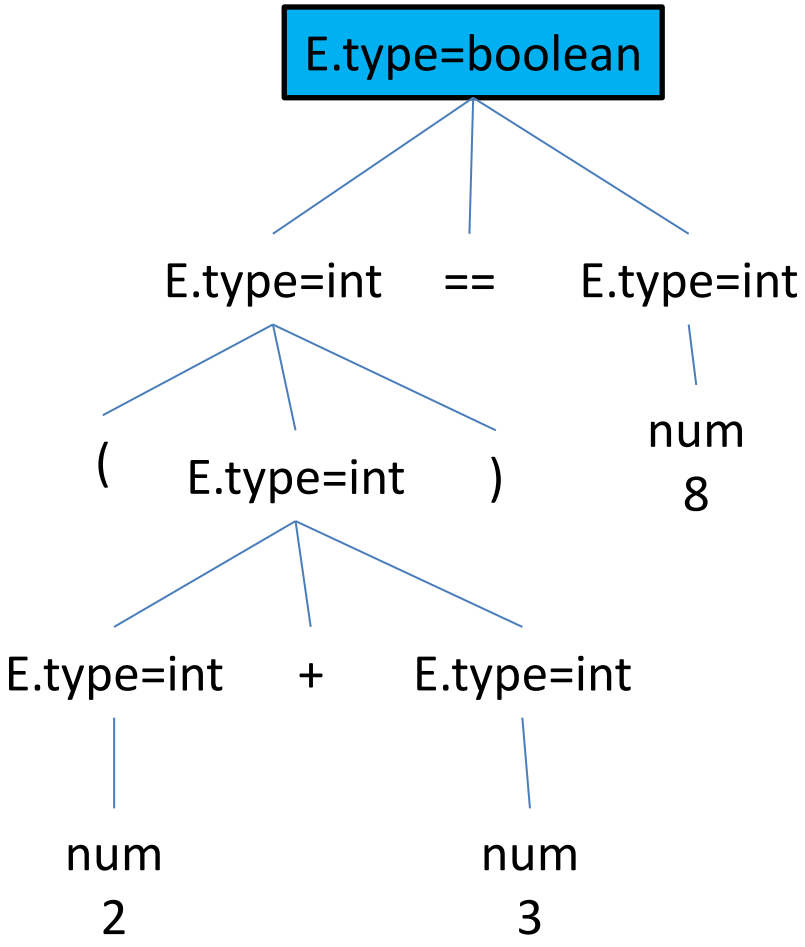
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 + 3) == 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

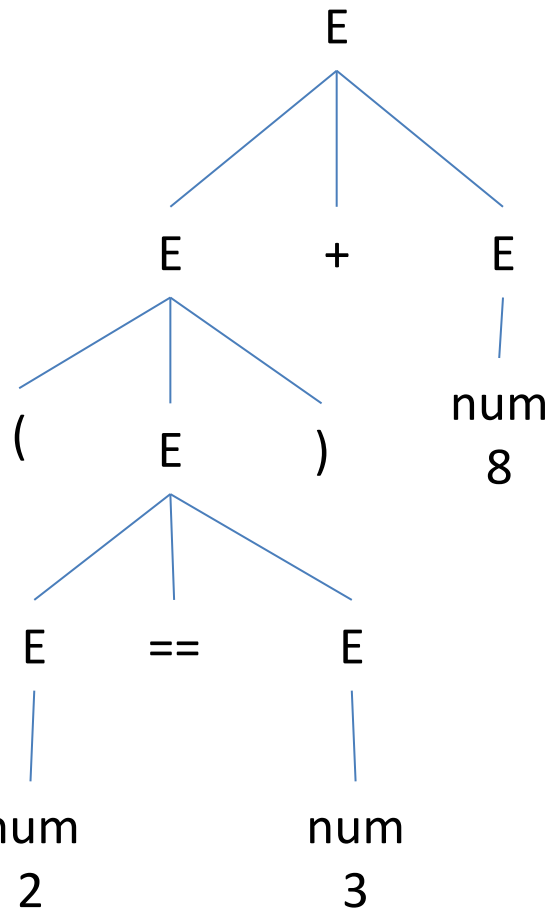
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

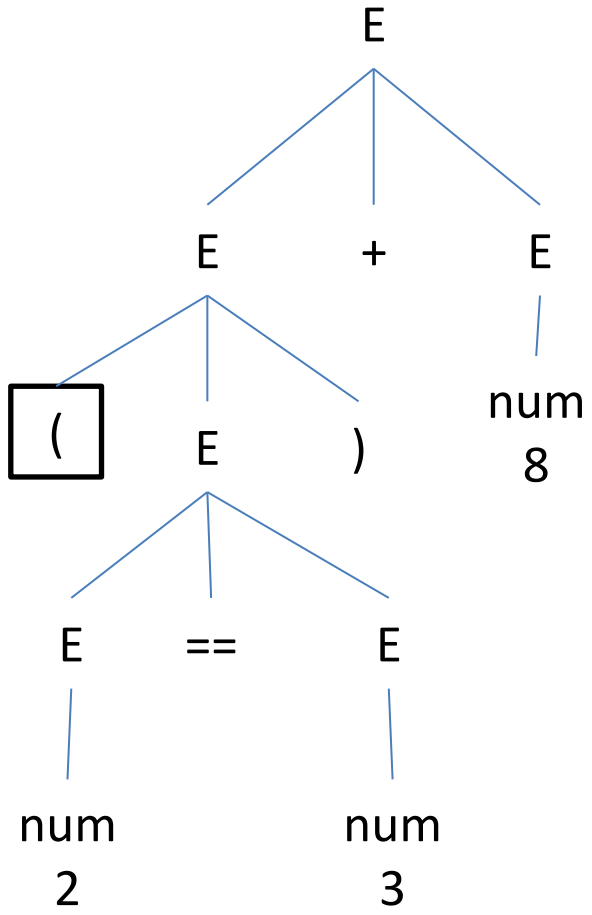
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int))
then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) &&
(E_1.type == int|boolean))
then E.type = boolean else error;}

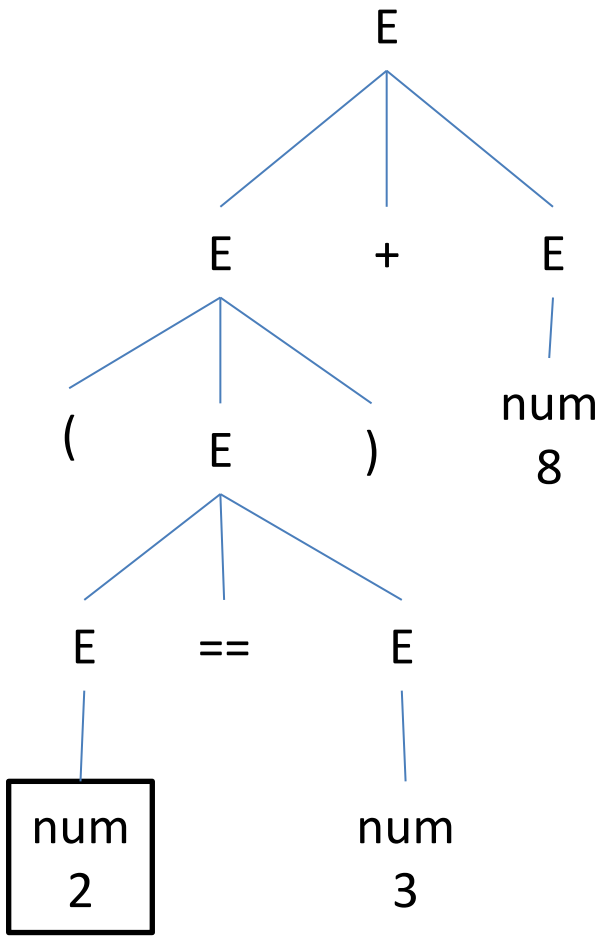
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

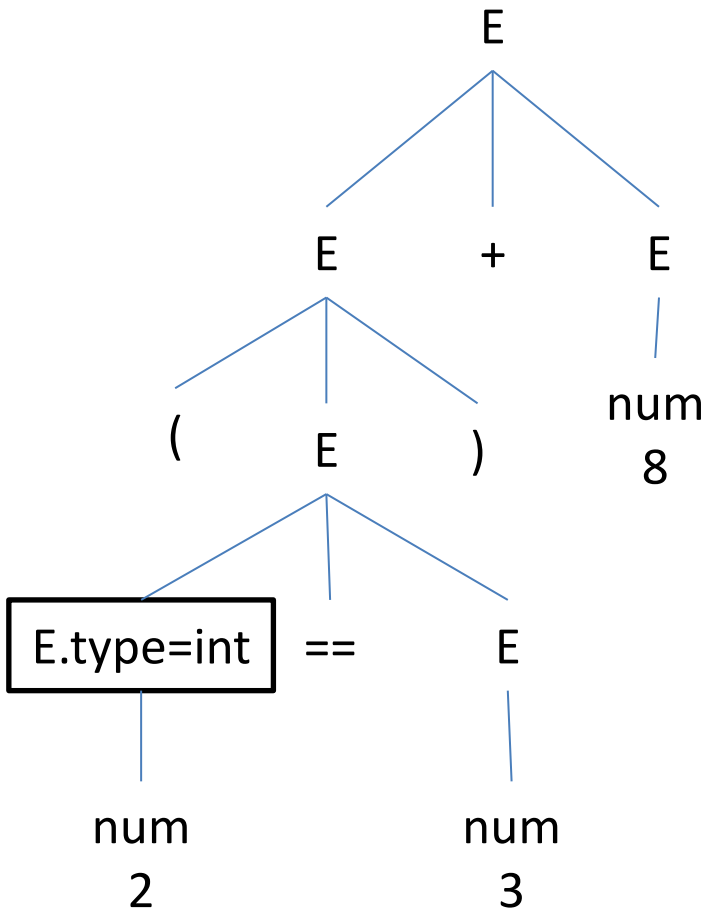
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

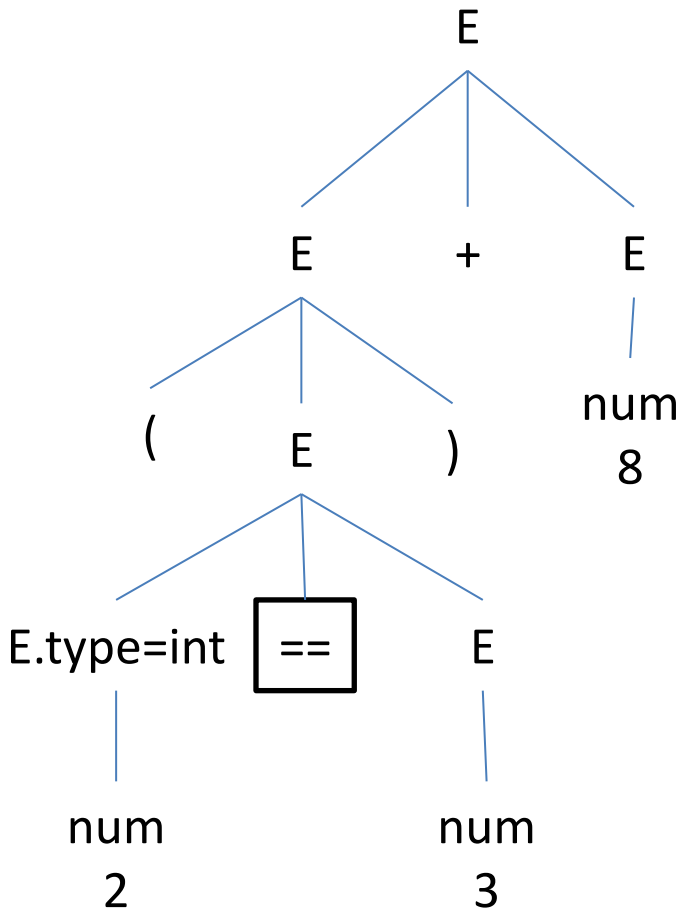
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

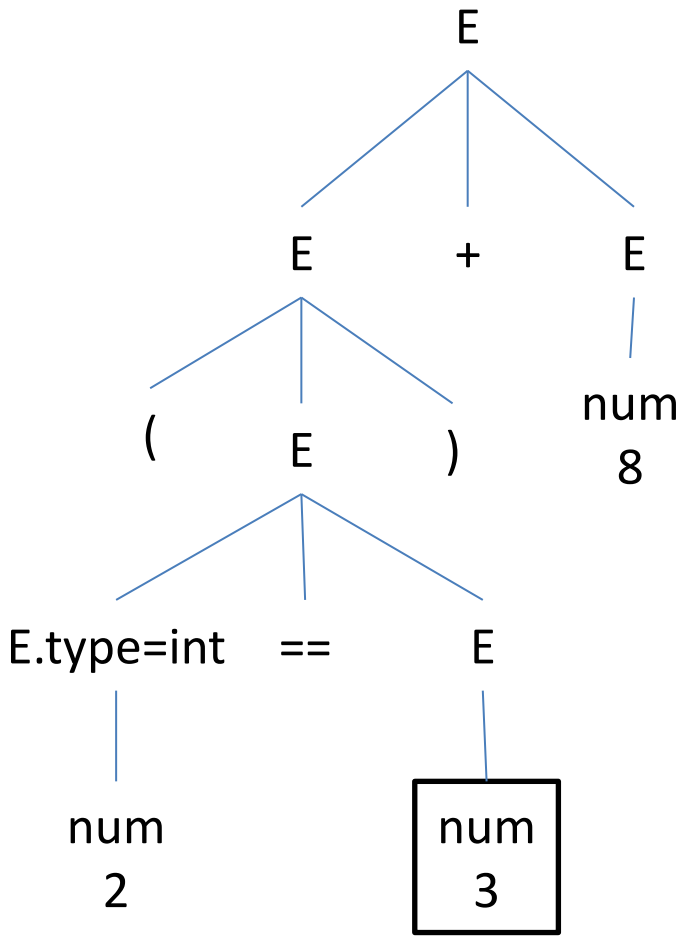
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

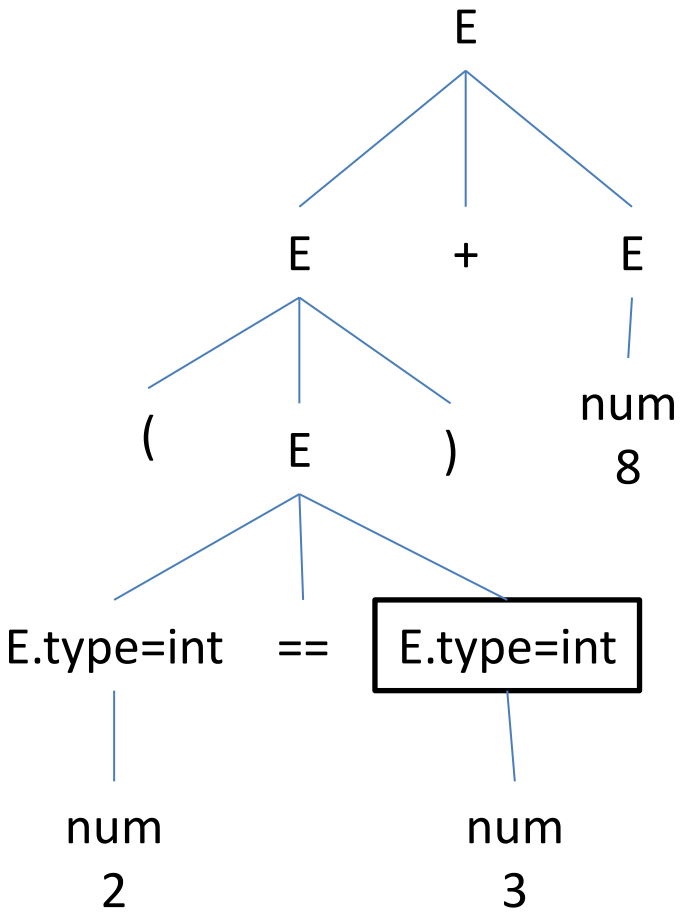
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

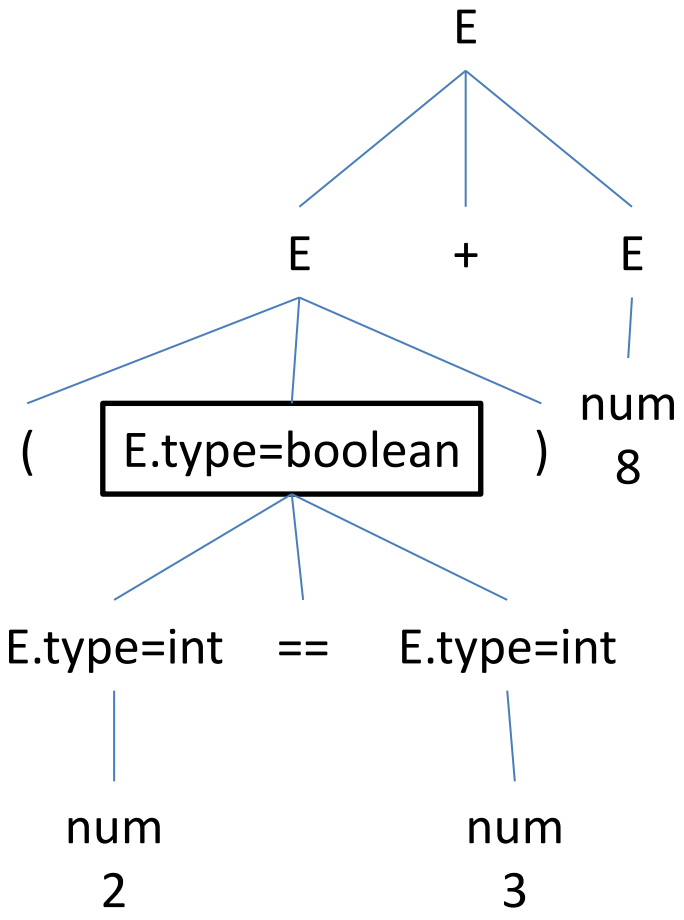
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

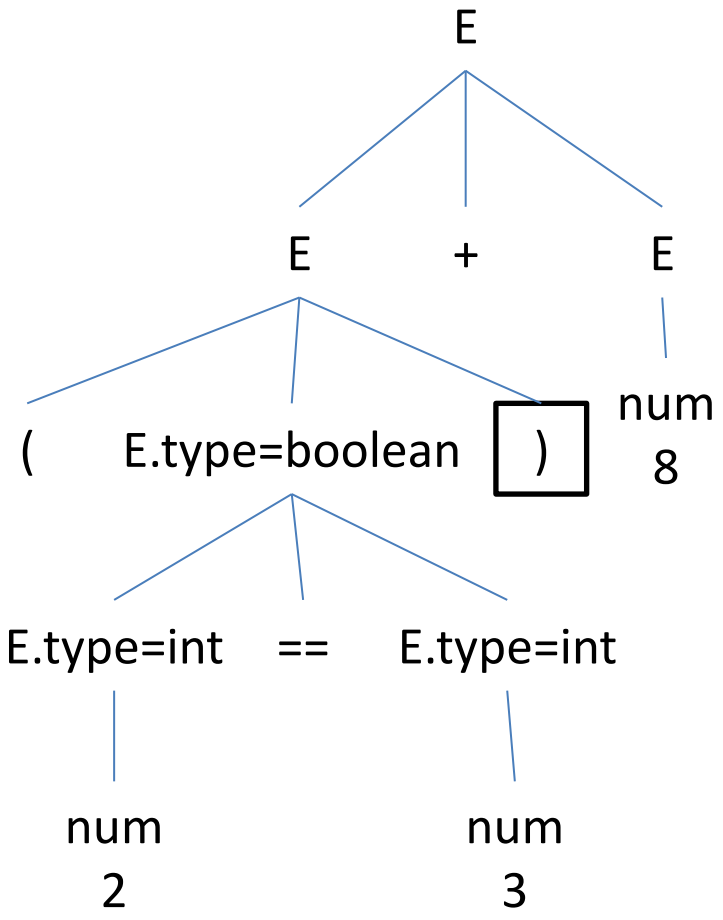
$E \rightarrow (E_1)$ {E.type = E_1.type}

$E \rightarrow$ num {E.type = int}

$E \rightarrow$ true {E.type = boolean}

$E \rightarrow$ false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

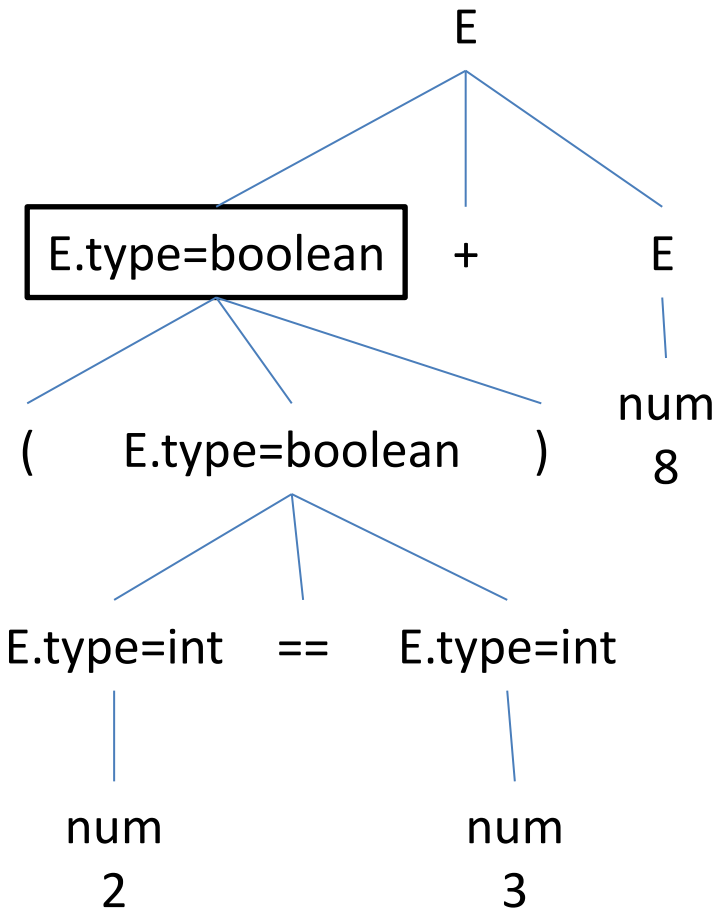
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

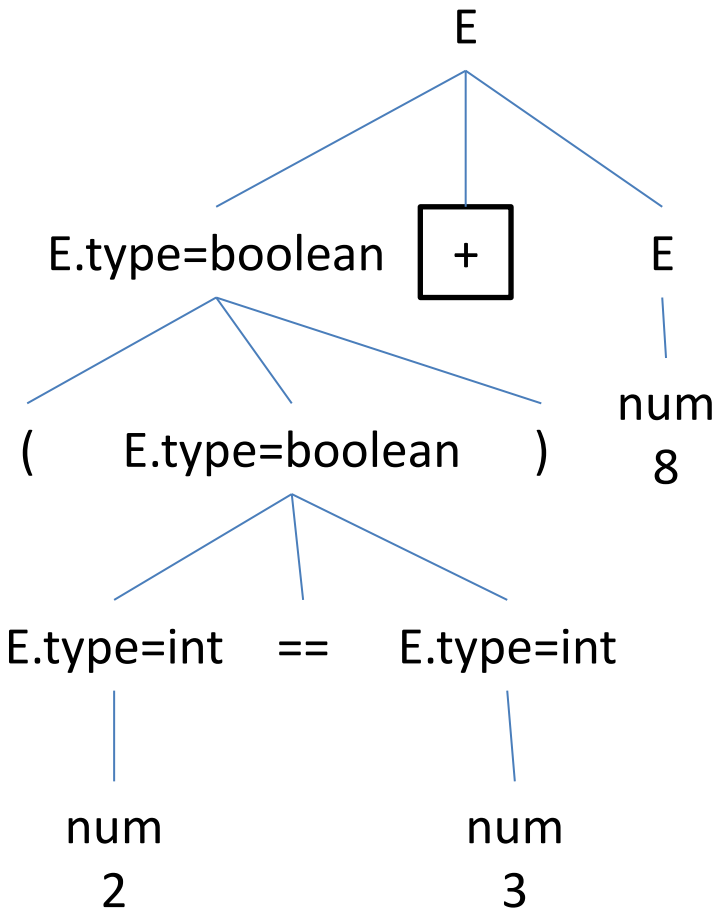
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

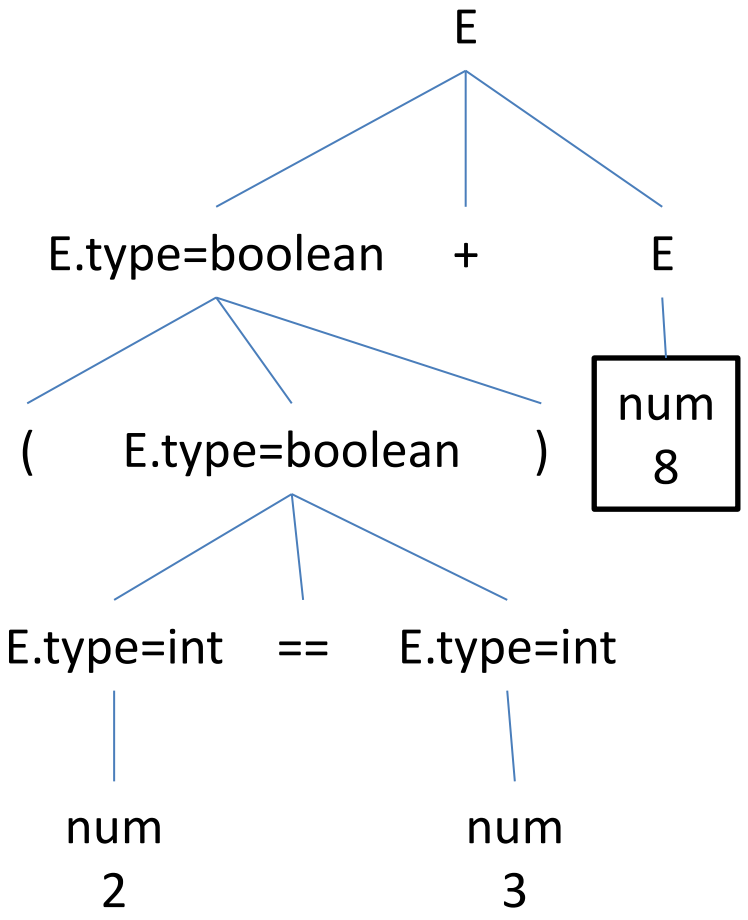
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

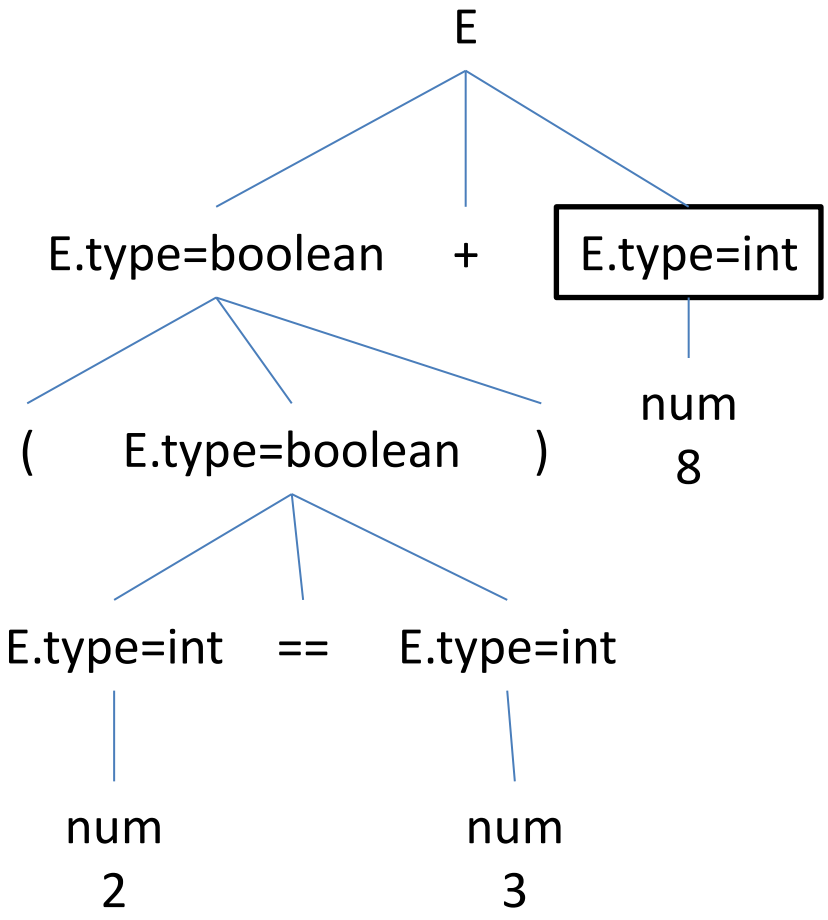
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8



Example 8

- SDT to check the type of an expression

$E \rightarrow E_1 + E_2$ {if((E_1.type == E_2.type) && (E_1.type == int)) then E.type = int else error;}

$E \rightarrow E_1 == E_2$ {if((E_1.type == E_2.type) && (E_1.type == int|boolean)) then E.type = boolean else error;}

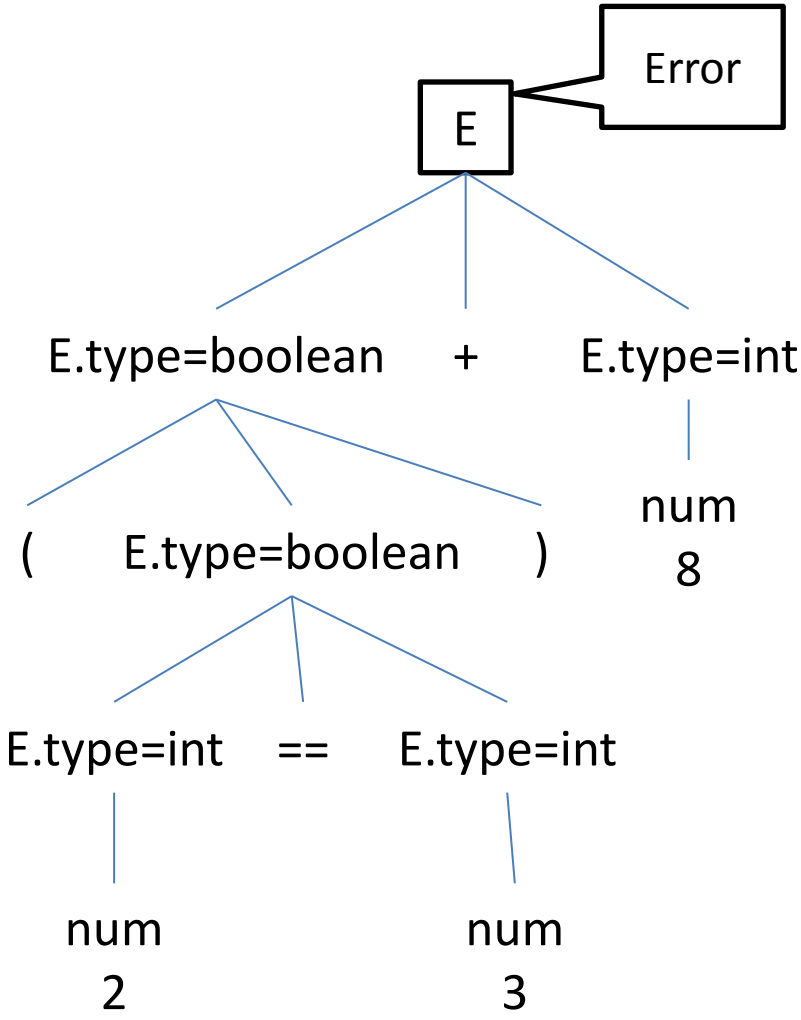
$E \rightarrow (E_1)$ {E.type = E_1.type}

E → num {E.type = int}

E → true {E.type = boolean}

E → false {E.type = boolean}

(2 == 3) + 8 — Error



S-attributed vs. L-attributed

1. S-attributed uses only synthesized attributes.
2. Semantic actions are placed at the right end of the productions.
3. Attributes are evaluated during bottom-up parsing.

1. L-attributed uses both inherited and synthesized attributes.
   - Inheritance is either from parent or left sibling.
2. Semantic action can be placed anywhere.
3. Attributes are evaluated by traversing the parse tree depth first, left to right.

Example

A→LM{L.i = f(A.i); M.i = f(L.s); A.s = f(M.s);}

A→QR {R.i = f(A.i); Q.i = f(R.i); A.s = f(Q.s);}

What type of SDT is this?

S-attributed or L-attributed?

Example

A→LM {L.i = f(A.i);	L's attribute is inherited from left side A
M.i = f(L.s);	M's attribute is inherited from left side L
A.s = f(M.s);}	A's attribute is synthesized from child M
A→QR {R.i = f(A.i);	R's attribute is inherited from left side A
Q.i = f(R.i);	Q's attribute is inherited from right side R
A.s = f(Q.s);}	A's attribute is synthesized from child Q

Example

A→LM {L.i = f(A.i);	L's attribute is inherited from left side A
M.i = f(L.s);	M's attribute is inherited from left side L
A.s = f(M.s);}	A's attribute is synthesized from child M

A→QR {R.i = f(A.i);	R's attribute is inherited from left side A
Q.i = f(R.i);	Q's attribute is inherited from right side R
A.s = f(Q.s);}	A's attribute is synthesized from child Q

Is it S-attributed? No because there are some inherited attributes.

Example

A→LM {L.i = f(A.i);	L's attribute is inherited from left side A
M.i = f(L.s);	M's attribute is inherited from left side L
A.s = f(M.s);}	A's attribute is synthesized from child M

A→QR {R.i = f(A.i);	R's attribute is inherited from left side A
Q.i = f(R.i);	Q's attribute is inherited from right side R
A.s = f(Q.s);}	A's attribute is synthesized from child Q

Is it S-attributed? No because there are some inherited attributes.

Is it L-attributed? No because there is inheritance from right side.

Thank You!!!

All The Best!!!