int x, y, z ;

the identifiers x, y, z will have type information associated with that.

Definition → Type Name ;

D → T N ;

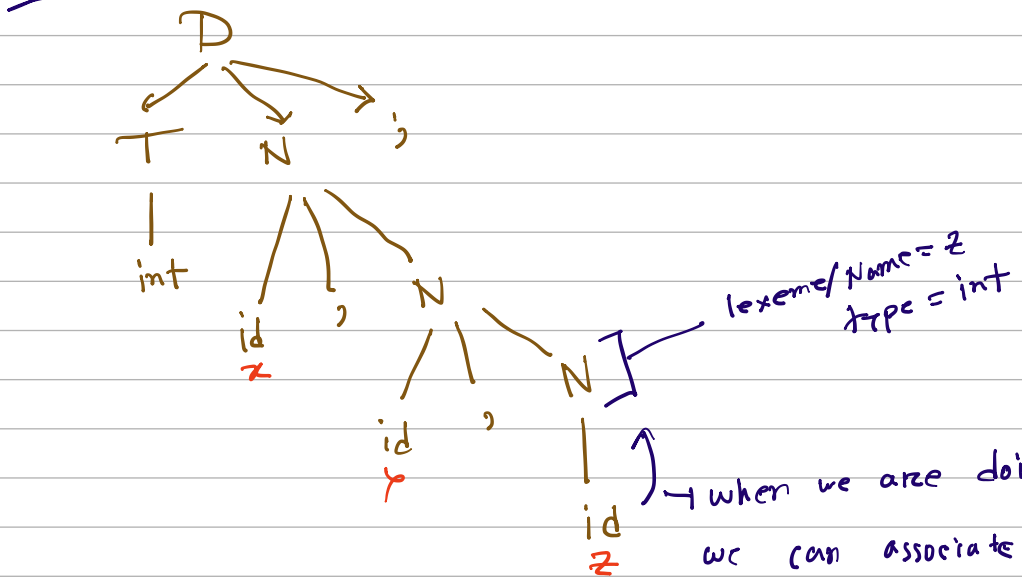
T → int

T → float

N → id , N

N → id

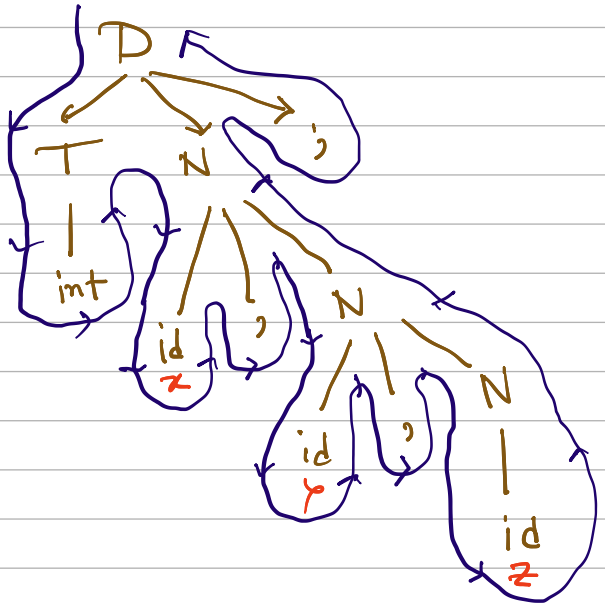
parse tree



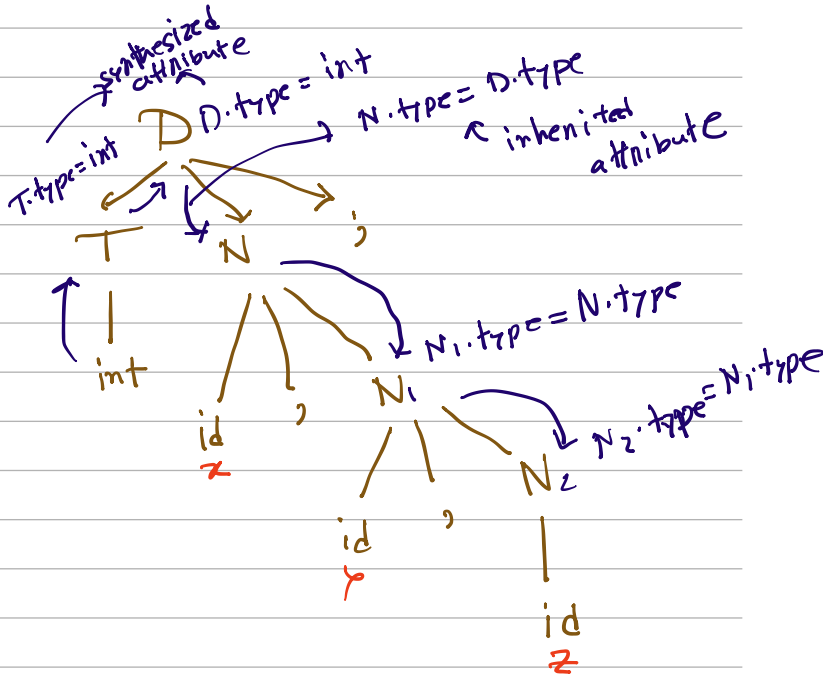
→ when we are doing this reduction we can associate name = z but we don't know the type.

we can construct a syntax tree and do a post order traversal.

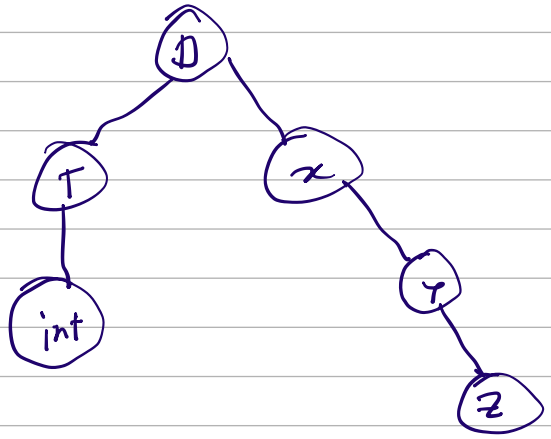
post order = left → right → parent traversal



type comes before the variable names. This



syntax tree



the traversal will be on this tree not parse tree.

* But in the parse tree we appointed D.type after seeing T but we did not see N yet.

$D \rightarrow T \cdot N ;$

int

{ D.type = T.type
N.type = D.type }

→ This one redundent

$D \to T \begin{Bmatrix} D.type = T.type \\ N.type = D.type \end{Bmatrix} N ;$ } $D \to T \{N.type = T.type\} N;$

as if earlier child is passing info to later children.

$T \to int \; \{T.type = int\}$

$T \to float \; \{T.type = float\}$

$N \to id \; , \{N_1.type = N.type\} \; N_1$

$N \to id$

LR Parsing

Pass information for attribute computation:

1. From children to parent → synthesized
2. From parent to children
3. From earlier children of a parent to later children

(2, 3) → inherited attribute

For attribute grammar when we follow all three rules the SDD rules are called L-attributed definitions

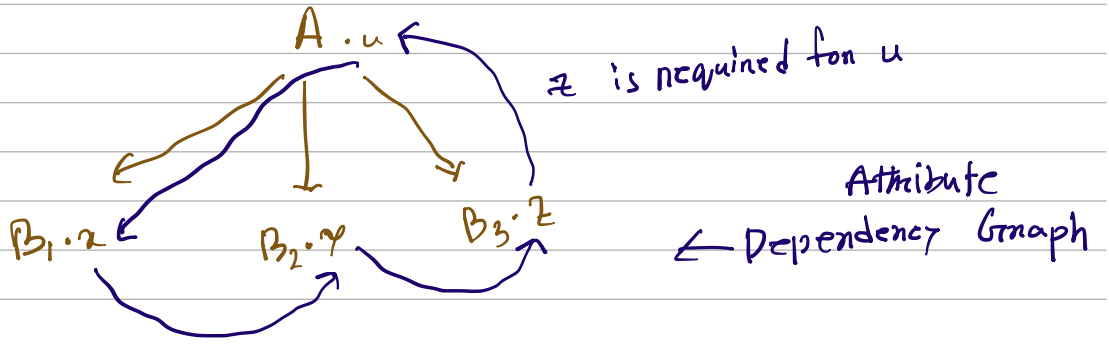
L ↓ left → going & passing info left to right.

If the grammar follow only the first rule then S-attributed definition.

When we either use S or L-attributed definition we can avoid circular dependencies.

what is circular dependency?

$A \to B_1 B_2 B_3$



but z needs T & y needs x but x needs u.

but without z we cannot have u.

Thus we are stuck.

Automated tools cannot detect this circular dependency, we need to draw & check if there's any.

Handling Array:

$D \rightarrow T\ id\ ;$

$T \rightarrow B\ C$

$B \rightarrow int$

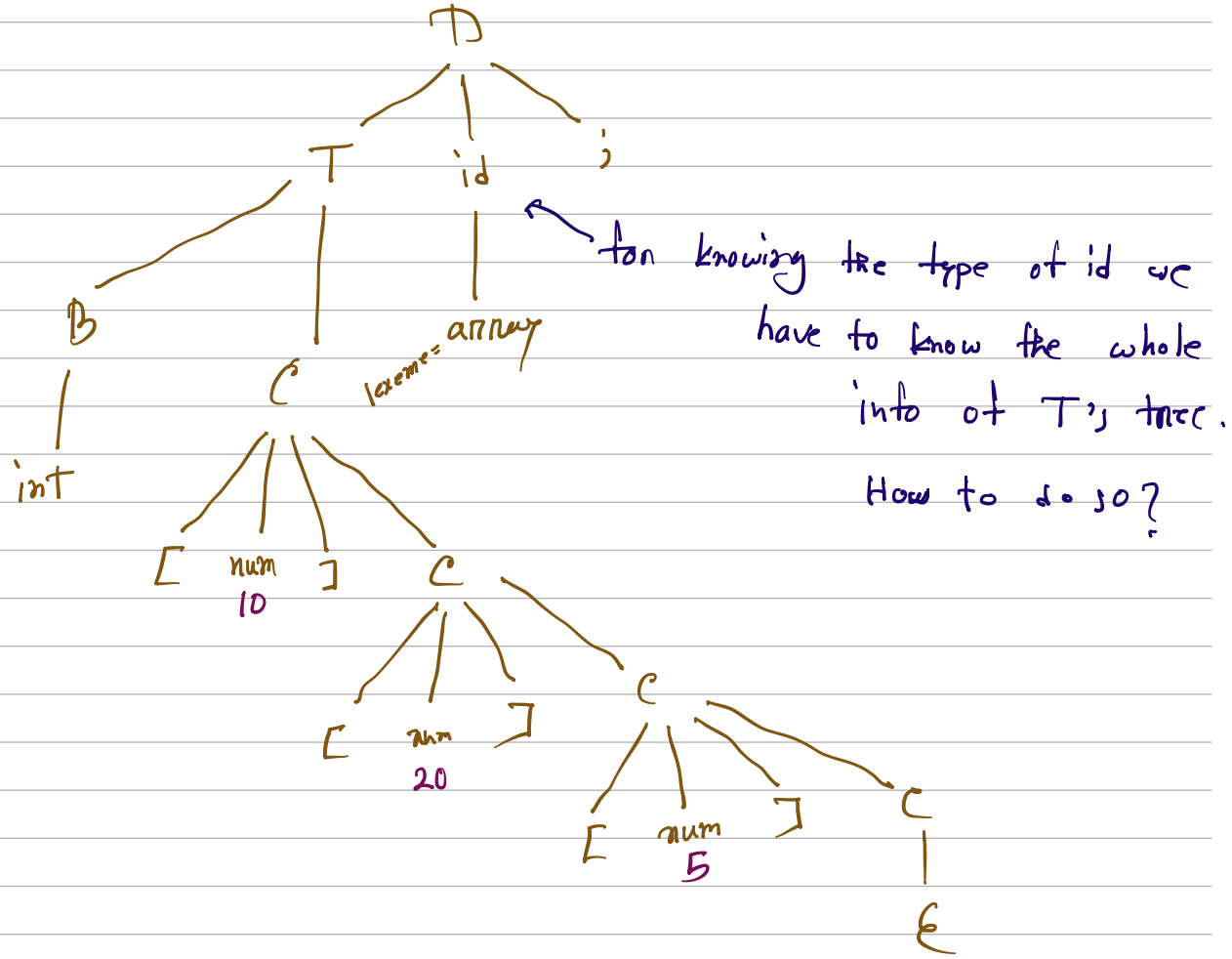
$B \rightarrow Float$

$C \rightarrow \epsilon$

$C \rightarrow [num]\ C_1 \Rightarrow C \rightarrow [E]\ C_1$ } For dynamic array

↑ Expression to calculate index

int [10] [20] [5] array ;



for knowing the type of id we have to know the whole 'into of T' tree.

How to do so?