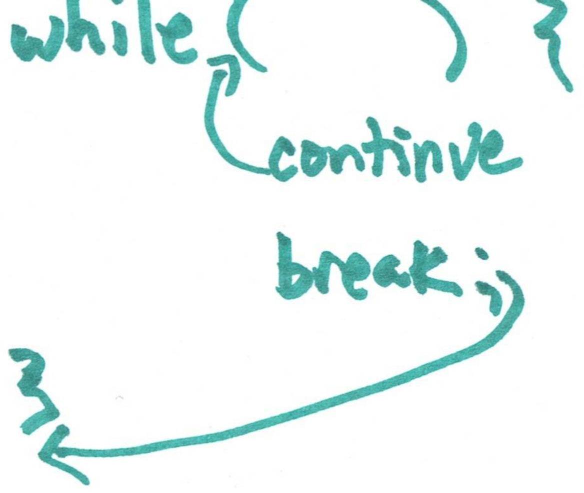
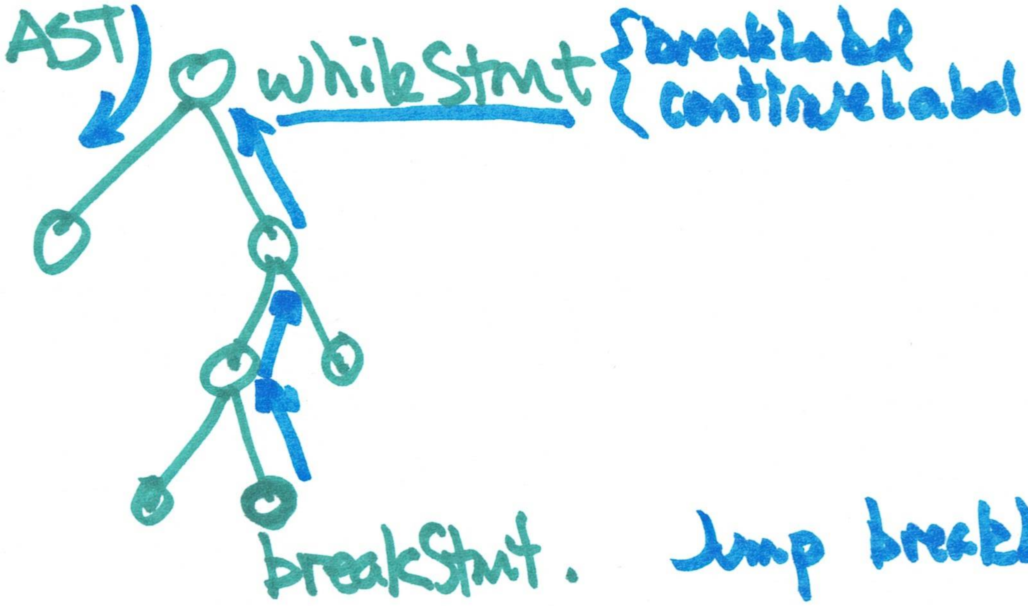
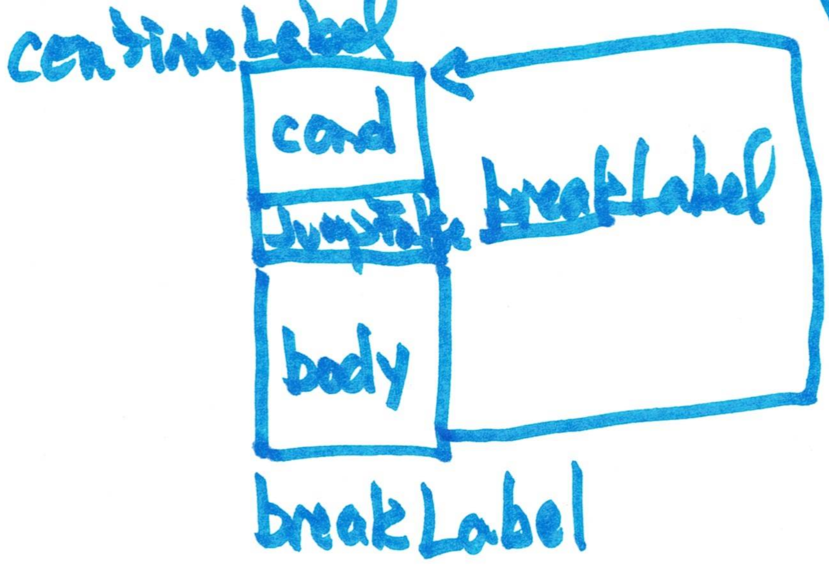




Jump breakLabel.

visitLeave



```
visitEnter(WhileStmtNode) {

    breakLabel = ~~new~~ Labeller.
    continueLabel = Labeller. —
    node.setBreakLabel(breakLabel
    node.setContinueLabel(continue
```

Attribute grammar $\left.\begin{array}{l}\\ \\ \end{array}\right\}$ grammar + rules.
Attributed grammar

$E_1 \rightarrow (E_2)$

$E_1.type = E_2.type$
$E_1.isConstant = E_2.isConstant$

---

$E_1 \rightarrow L + E_2$

$E_1.type =$ if ($L.type ==$ <u>int</u> && $E_2.type ==$ int)
then <u>int</u> else <u>float</u> ;

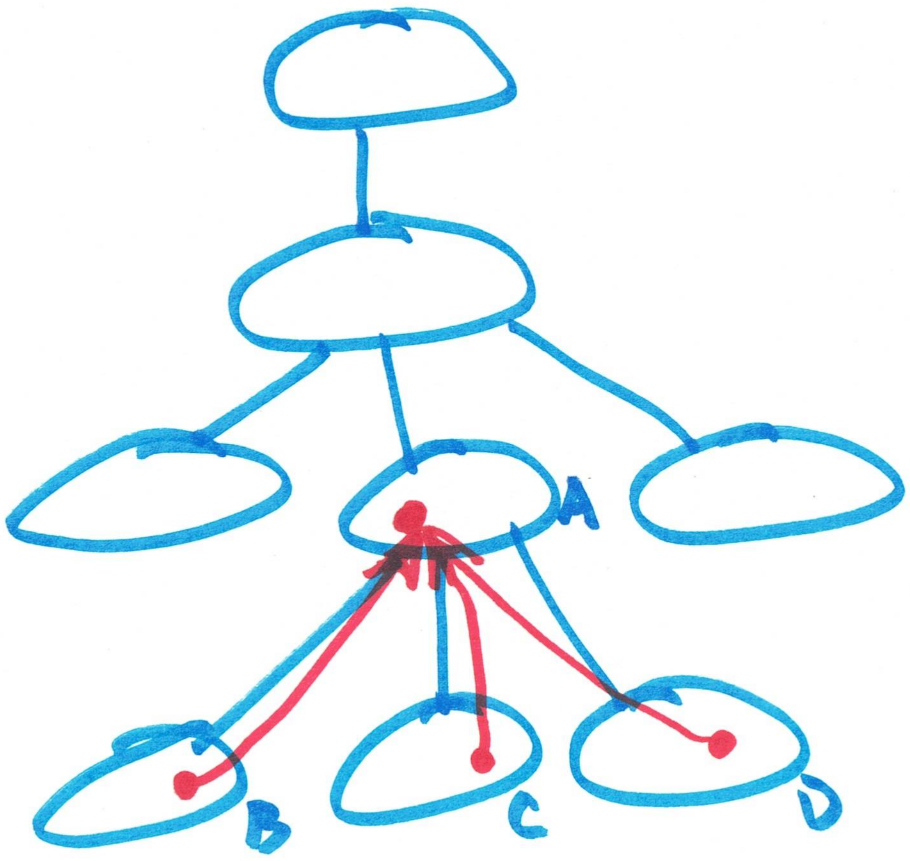
$E \rightarrow L$

$E.type = L.type$
$E.isConstant = L.isConstant$

$L \rightarrow (E)$

$L.type = E.type$
$L.isConst = E.isConst$

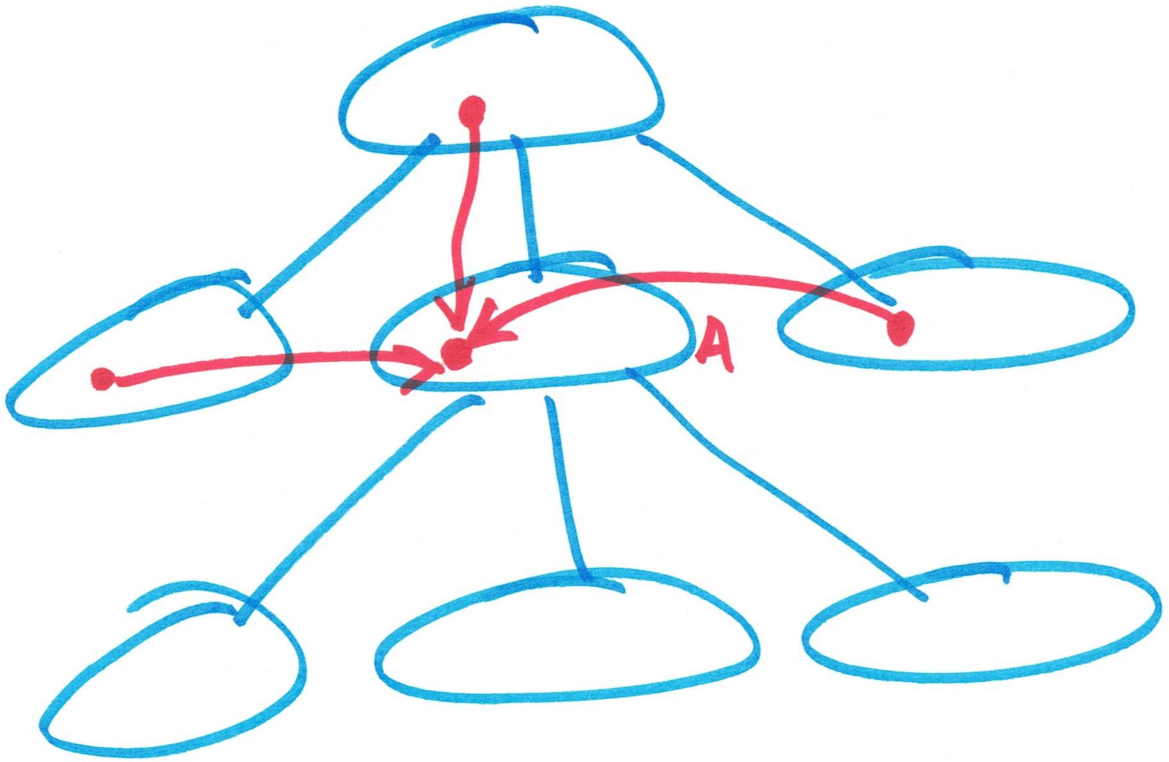
| | |
|---|---|
| $L \rightarrow v$ | L.type = v.type |
| | v.type = v.getBinding().getType() |
| | L.isConst = $\underline{\text{false}}$ |
| $L \rightarrow ic$ | L.type = ic.type |
| | ic.type = $\underline{\text{int}}$ |
| | L.isConst = $\underline{\text{true}}$ |
| $L \rightarrow fc$ | L.type = fc.type |
| | fc.type = $\underline{\text{float}}$ |
| | L.isConst = $\underline{\text{true}}$ |



A → B C D

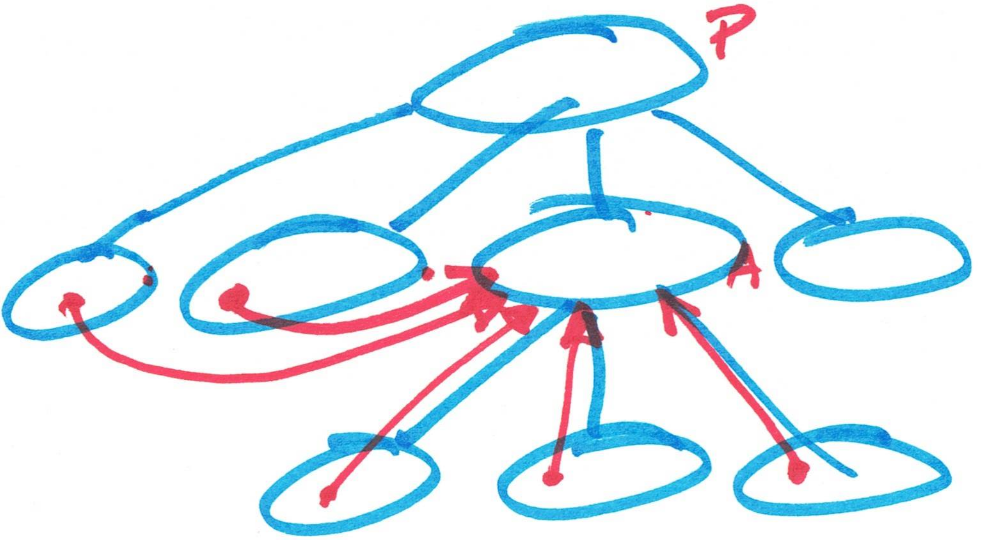
A.attr :=
f(B.$a_1$, C.$a_2$,
D.$a_3$);

A.attr. is <u>synthesized</u>
or <u>synthetic</u>.



A.attr is <u>inherited</u> attribute

Attributed Grammar is called S-attributed if every attribute is synthetic.



L-attributed if every attribute depends on children attributes and left-sibling attributes

If grammar is not L-attributed, use a more complex evaluation scheme than postorder. Generally compiler will do a topological sort on the attribute digraph.