



Compilers

Implementing Type Checking

- COOL type checking can be implemented in a single traversal over the AST
- Type environment is passed down the tree
 - From parent to child
- Types are passed up the tree
 - From child to parent

$$\left[\frac{O, M, C \vdash e_1 : \underline{\text{Int}} \quad O, M, C \vdash e_2 : \underline{\text{Int}}}{O, M, C \vdash e_1 + e_2 : \underline{\text{Int}}} [\text{Add}] \right]$$

O, M, C
 TypeCheck(Environment, $e_1 + e_2$) = {
 → T_1 = TypeCheck(Environment, e_1);
 T_2 = TypeCheck(Environment, e_2);
 Check T_1 == T_2 == Int;
 return Int; }

$$\frac{
 \frac{
 \underline{O} \vdash e_0 : \underline{T_0} \quad
 \underline{O[T/x]} \vdash e_1 : \underline{T_1} \quad \leftarrow
 }{
 \underline{T_0} \leq \underline{T}
 }
 }{
 \underline{O} \vdash \text{let } \underline{x:T} \leftarrow \underline{e_0} \text{ in } \underline{e_1} : \underline{T_1}
 } \quad [\text{Let-Init}]$$

TypeCheck(Environment, let x:T ← e₀ in e₁) = {
 T₀ = TypeCheck(Environment, e₀);
 T₁ = TypeCheck(Environment.add(x:T), e₁);
 Check subtype(T₀, T);
 return T₁}