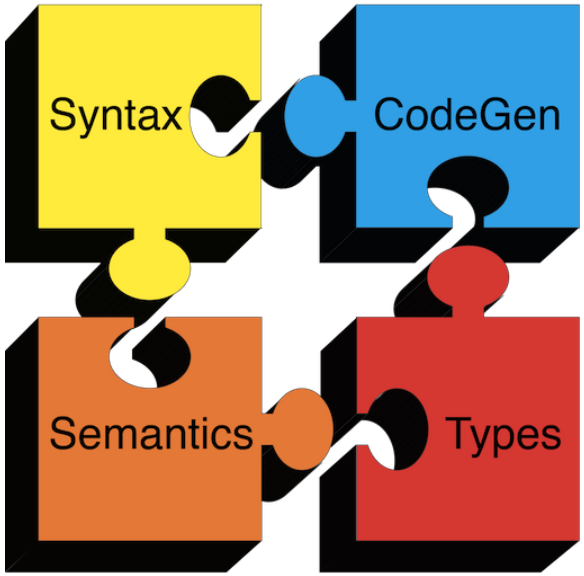


Compilers

Error Recovery

Error Recovery

- As with parsing, it is important to recover from type errors
- Detecting where errors occur is easier than in parsing
  - There is no reason to skip over portions of code
- The problem:
  - What type is assigned to an expression with no legitimate type?
  - This type will influence the typing of the enclosing expression

Error Recovery

- Assign type Object to ill-typed expressions

  let y : Int ← x + 2 in y + 3

⇒ a workable solution but with cascading errors

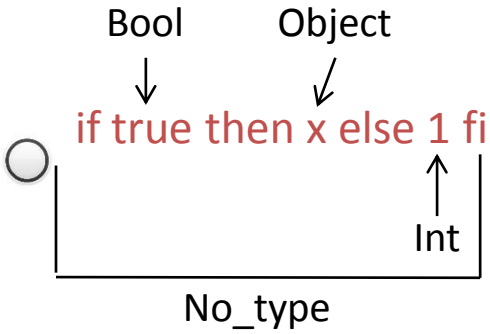
Error Recovery

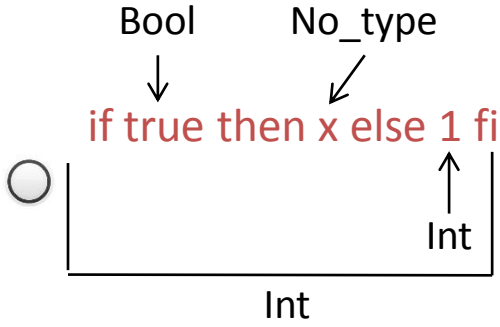
- Introduce a new type No_type for use with ill-typed expressions
- Define No_type $\leq$ C for all types C
- Every operation is defined for No_type
  - With a No_type result

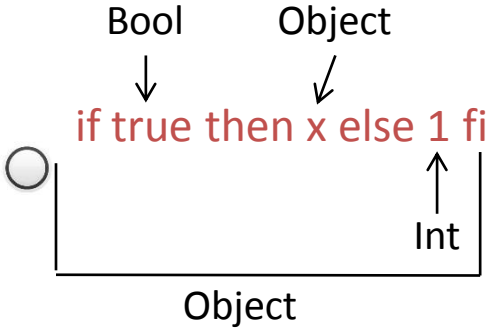
let y : Int $\leftarrow$ x + 2 in y + 3

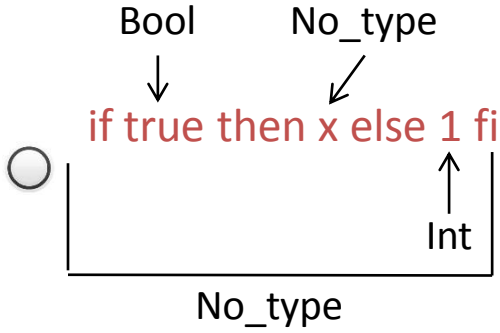
Error Recovery

Choose the correct labeling of types for the code fragment, using No_type as described in the video. Assume that x is not defined.









Error Recovery

- A “real” compiler would use something like No_type
- However, there are some implementation issues
  - The class hierarchy is not a tree anymore
- The Object solution is fine in the course project