Types and evaluation

• Why is a type system useful?

 \rightarrow It can rule out ill-formed programs before we run them

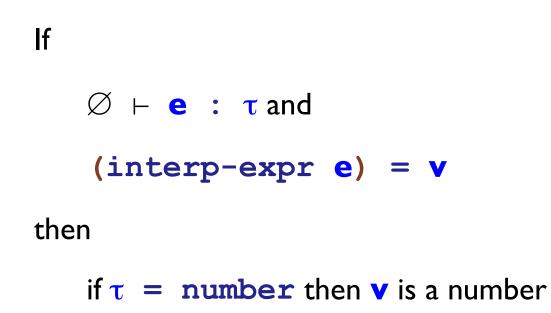
• What information can a type system give us?

 \rightarrow The type of data the program should produce as a result

• What is the relationship between:

 \rightarrow v should be consistent with τ

• We'd like types to tell us something useful about the behavior of our program at run-time



if $\tau = (\tau_1 \rightarrow \tau_2)$ then **v** is 'procedure

- With type soundness, our types accurately predict the kind of data we'll get when we run our program

 Guaranteed
- Without type soundness, may get bogus predictions

 So can't rely on it
 Invitation for bugs, security vulnerabilities, yikes
- Formal property, can be proven mathematically
 - Starting from typing rules
 - Bugs may creep in as you go from rules to code!

Not all type systems used in practice are sound!

- Standard ML: proven sound
- Haskell: subsets have been proven sound
 Whole type system proven sound at one point
 But constantly evolves, so may be out of date
- Rust: proven sound, at least a subset (IIRC)
- Java: has soundness holes, but mostly hangs together
 O But soundness holes are enough for security holes!
- C: Iol, what's soundness