Type Soundness

Types and evaluation

- Why is a type system useful?

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

- What information can a type system give us?

  → The type of data the program should produce as a result

- What is the relationship between:

$$\Gamma \vdash \mathbf{e} : \tau$$

and

`interp-expr : e -> v`

→ **v** should be consistent with τ

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

Type Soundness

If

$\varnothing \vdash$ **e** : τ

then

`(interp-expr e)` = **v** and

if τ = `number` then **v** is a num

if τ = $(\tau_1$ `->` $\tau_2)$ then **v** is `'procedure`

Type Soundness

If we *only* allow programs such that

```
(interp-expr e) = v
```

```
{+ 5 false}
```

We'd like to rule out things like this, and we do.

Type Soundness

If we *only* allow programs such that

```
(interp-expr e) = v
```

```
{/ 5 ...}
```

We'd probably like to allow this.

But what if `...` evaluates to 0?

We're also forced to rule out programs that don't terminate, or may not terminate.

But neither of these really are "type" errors.

That's too conservative.

Type Soundness

If

$\varnothing \vdash$ **e : τ** and

`(interp-expr e) = v`

then

if τ = `num` then **v** is a num

if τ = `(τ1 -> τ2)` then **v** is `'procedure`

- The condition on interpreted values is now a premise
- This allows the programs we want to allow, but considerably weakens the statement of type soundness

Type Soundness

- 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!

Type Soundness

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    - But constantly evolves, so may be out of date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
    - But soundness holes are enough for security holes!
- C: lol, what's soundness