LET'S DO THIS PART WITH SLIDES...

(Look at the Agda code first)

```
NonZero (head xs) → NonZero (sum xs)
```

WORKING THROUGH AN EXAMPLE

Create metavariables for generalizable variables

```
?l  : Level
?A  : Set ?l
?m  : Nat
?xs : Vec ?A ?m
```

```
NonZero (head ?xs) → NonZero (sum ?xs)
```

WORKING THROUGH AN EXAMPLE

Type check

```
?l  : Level      := lzero
?A  : Set ?l     := Nat
?m  : Nat        := suc ?n
?xs : Vec ?A ?m
?n  : Nat
```

```
NonZero (head ?xs) → NonZero (sum ?xs)
```

WORKING THROUGH AN EXAMPLE

Type check

```
?l  : Level      := lzero
?A  : Set ?l     := Nat
?m  : Nat        := suc ?n
?xs : Vec Nat (suc ?n)
?n  : Nat
```

```
NonZero (head ?xs) → NonZero (sum ?xs)
```

WORKING THROUGH AN EXAMPLE

Collect and sort the unsolved metavariables

```
?n  : Nat
?xs : Vec Nat (suc ?n)
```

... and generalize

```
    {n : Nat} {xs : Vec A n} →
NonZero (head xs) → NonZero (sum xs)
```

THAT WAS EASY
(TOO EASY)

- When the type checker gives you a type checked term it expects you to treat it like one!

- In particular

```
⊢ NonZero (head ?xs) → NonZero (sum ?xs)
```

```
(n : Nat)(xs : Vec A n) ⊢
NonZero (head xs) → NonZero (sum xs)
```

- Valid moves are: solve meta, apply substitution, formation rules, ...

DOING IT PROPERLY

Create metavariables for generalizable variables

```
?G  : Set ?k
?l  : ?G → Level
?A  : (g : ?G) → Set (?l g)
?m  : ?G → Nat
?xs : (g : ?G) → Vec (?A g) (?m g)
```

```
(g : ?G) ⊢ NonZero (head (?xs g)) → NonZero (sum (?xs g))
```

DOING IT PROPERLY

Type check

```
?G  : Set?k
?l  := λ _ → lzero
?A  := λ _ → Nat
?m  := λ g → suc (?n g)
?xs : (g : ?G) → Vec Nat (suc (?n g))
?n  : ?G → Nat
```

```
(g : ?G) ⊢ NonZero (head (?xs g)) → NonZero (sum (?xs g))
```

DOING IT PROPERLY

Collect and sort the unsolved metavariables

```
?n  : ?G → Nat
?xs : (g : ?G) → Vec Nat (suc (?n g))
```

Solve ?G

```
?G := (n : Nat) × Vec Nat (suc n)
```

Solve metas with projections from g

```
?n  := λ g → fst g
?xs := λ g → snd g
```

Result:

```
          (g : (n : Nat) × Vec Nat n) ⊢
NonZero (head (snd g)) → NonZero (sum (snd g))
```

DOING IT PROPERLY

```
(g : (n : Nat) × Vec Nat n) ⊢
NonZero (head (snd g)) → NonZero (sum (snd g))
```

Create a substitution **σ**

```
(n : Nat)(xs : Vec Nat n) ⊢ σ : (g : (n : Nat) × Vec Nat n)
σ = [g := (n , xs)]
```

Apply **σ**

```
(n : Nat)(xs : Vec Nat n) ⊢
NonZero (head xs) → NonZero (sum xs)
```

...and generalize

```
⊢ {n : Nat}{xs : Vec Nat n} →
NonZero (head xs) → NonZero (sum xs)
```

RELATED WORK

- Twelf
  Generalize over all unbound variables

- Idris
  "Any name beginning with a lower case letter which appears as a parameter or index in a type declaration, which is not applied to any arguments, will *always* be automatically bound as an implicit argument." -- Idris User Manual

- Coq
  Implicit contexts

- GHC
  With DataKinds GHC has to do very similar implicit generalization

QUESTIONS?

- Released with Agda 2.6.0
- Some small bugs to iron out
- People are using it in anger