

# A Second Look at Overloading

```

inst eq : Nat → Nat → Bool
  eq 0      0      = True
  eq (S x) (S y) = eq x y
  eq _      _      = False

inst eq : (eq : a → a → Bool) ⇒ [a] → [a] → Bool
  eq []      []      = True
  eq (x:xs) (y:ys)    = eq x y && eq xs ys
  eq _      _      = False

let isEq = eq [0] [0]

```

Pseudocode — Example

$e ::= x$  $| \lambda x. e$  $| e_1 e_2$  $| \mathbf{let} \ x = e_2 \ \mathbf{in} \ e_1$  $\tau ::= \alpha$  $| \tau_1 \rightarrow \tau_2$  $\sigma ::= \tau$  $| \forall \alpha. \sigma$ 

```
let cons ::  $\forall a. a \rightarrow [a] \rightarrow [a]$  =  $\lambda x. \lambda lst. x : lst$  in ..
```

```
let evil ::  $(\forall a. a \rightarrow a) \rightarrow \mathbf{Unit}$  =  $\lambda id. id \ 42; id \ \text{"foo"}; in ..$ 
```

```
 $(\lambda id. .. (id \ 42) .. (id \ \text{"foo"}) ..) (\lambda x. x)$ 
```

$e ::= x$  $| \lambda x. e$  $| e_1 e_2$  $| \mathbf{let} \ x = e_2 \ \mathbf{in} \ e_1$  $p ::= e$  $| \mathbf{inst} \ o : \sigma_T = e \ \mathbf{in} \ p$  $\tau ::= \alpha$  $| \tau_1 \rightarrow \tau_2$  $\pi_\alpha ::= o_i : \alpha \rightarrow \tau_i \quad i \in \mathbb{N}$  $\sigma ::= \tau$  $| \forall \alpha. \pi_\alpha \Rightarrow \sigma$ 

```
inst eq : Nat → Nat → Bool = λx. λy. .. in
inst eq : ∀a. (eq : a → a → Bool) ⇒ [a] → [a] → Bool = .. in
eq [0] [0]
```

$$\text{(INST)} \quad \frac{\Gamma \vdash e : \sigma_T \quad \Gamma, o : \sigma_T \vdash p : \sigma \quad \forall (o : \sigma_{T'}) \in \Gamma : T \neq T'}{\Gamma \vdash \mathbf{inst} \ o : \sigma_T = e \ \mathbf{in} \ p : \sigma}$$

$$\text{(\forall I)} \quad \frac{\Gamma, \pi_\alpha \vdash e : \sigma \quad \text{fresh } \alpha}{\Gamma \vdash e : \forall \alpha. \pi_\alpha \Rightarrow \sigma}$$

$$\text{(\forall E)} \quad \frac{\Gamma \vdash e : \forall \alpha. \pi_\alpha \Rightarrow \sigma \quad \Gamma \vdash [\tau / \alpha] \pi_\alpha}{\Gamma \vdash e : [\tau / \alpha] \sigma}$$

System 0 — Typing

$$\Gamma = \{\text{eq} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B},$$

$$\text{eq} : \forall \alpha. (\text{eq} : \alpha \rightarrow \alpha \rightarrow \mathbb{B}) \Rightarrow [\alpha] \rightarrow [\alpha] \rightarrow \mathbb{B}\}$$

$$\frac{\text{eq} : \forall \alpha. (\text{eq} : \alpha \rightarrow \alpha \rightarrow \mathbb{B})}{\Rightarrow [\alpha] \rightarrow [\alpha] \rightarrow \mathbb{B} \in \Gamma}$$

$$\frac{\Gamma \vdash \text{eq} : \forall \alpha. (\text{eq} : \alpha \rightarrow \alpha \rightarrow \mathbb{B})}{\Rightarrow [\alpha] \rightarrow [\alpha] \rightarrow \mathbb{B}} \quad \frac{\text{eq} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B} \in \Gamma}{\Gamma \vdash \text{eq} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B}}$$

$$\frac{\Gamma \vdash \text{eq} : [\mathbb{N}] \rightarrow [\mathbb{N}] \rightarrow \mathbb{B} \quad \Gamma \vdash [0] : [\mathbb{N}] \quad \dots}{\Gamma \vdash \text{eq} [0] : [\mathbb{N}] \rightarrow \mathbb{B}} \quad \frac{\dots}{\Gamma \vdash [0] : [\mathbb{N}]}$$

$$\Gamma \vdash \text{eq} [0] [0] : \mathbb{B}$$

System 0 — Constraint Solving

$$\begin{aligned}
& \llbracket \text{inst } eq : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{B} = e_1 \text{ in} \\
& \quad \text{inst } eq : \forall \alpha. (eq : \alpha \rightarrow \alpha \rightarrow \mathbb{B}) \Rightarrow [\alpha] \rightarrow [\alpha] \rightarrow \mathbb{B} = e_2 \text{ in} \\
& \quad eq \ [0] \ [0] \rrbracket_{\emptyset} \\
= & \llbracket \text{inst } eq : \forall \alpha. (eq : \alpha \rightarrow \alpha \rightarrow \mathbb{B}) \Rightarrow [\alpha] \rightarrow [\alpha] \rightarrow \mathbb{B} = .. \text{ in} \\
& \quad eq \ [0] \ [0] \rrbracket_{\{eq := \lambda x. \text{ if } x \text{ is } \mathbb{N} \text{ then } \llbracket e_1 \rrbracket x\}} \\
= & \llbracket eq \ [0] \ [0] \rrbracket_{\{eq := \lambda x. \text{ if } x \text{ is List then } \llbracket e_2 \rrbracket x \text{ else } \lambda x. \text{ if } x \text{ is } \mathbb{N} \text{ then } \llbracket e_1 \rrbracket x\}} \\
= & (\lambda x. \text{ if } x \text{ is } [\alpha] \text{ then } \llbracket e_2 \rrbracket x \text{ else } \lambda x. \text{ if } x \text{ is } \mathbb{N} \text{ then } \llbracket e_1 \rrbracket x) \ [0] \ [0] \\
= & \llbracket e_2 \rrbracket \ [0] \ [0]
\end{aligned}$$

```
inst eq : Nat → Nat → Bool
  = λ.. in
inst eq : ∀a. (eq : a → a → Bool) ⇒ [a] → [a] → Bool
  = λ.. in
eq [0] [0]
```

System 0

```
let eq0 :: Nat → Nat → Bool
  = λ.. in
let eq1 :: ∀a. (a → a → Bool) → [a] → [a] → Bool
  = λeq0. λ.. in
eq1 eq0 [0] [0]
```

Hindley Milner

System 0 — Translation to Hindley Milner



```

let max :: ∀β. (gte : β → β → Bool) ⇒
           ∀α. (α ≤ {key: β}) ⇒ α → α → α
    = λx. λy. if gte x.key y.key then x else y in
max {field: "a", key: 1} {field: "b", key: 2}

```

## Records + Subtyping

```

inst field : ∀α. ∀β. R0 α β → α = λR0 x y. x in
inst key   : ∀α. ∀β. R0 α β → β = λR0 x y. y in
let max :: ∀β. (gte : β → β → Bool) ⇒
           ∀α. (key : α → β) ⇒ α → α → α
    = λx. λy. if gte (key x) (key y) then x else y in
max (R0 "a" 1) (R0 "b" 2)

```

## System 0

System 0 — Relationship with Record Typing

# Repository

[github.com/Mari-W/pop1](https://github.com/Mari-W/pop1)

# References

- [A Second Look at Overloading](#) 1995  
Martin Odersky, Philip Wadler, Martin Wehr
- [A Theory of Type Polymorphism in Programming](#) 1978  
Hindley Milner