A Second Look at Overloading

Marius Weidner – Chair of Programming Languages – Seminar '22

```
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

$$
\begin{aligned}
e ::= \;& x \\
\mid\;& \lambda x.\ e \\
\mid\;& e_1\ e_2 \\
\mid\;& \textbf{let}\ x = e_2\ \textbf{in}\ e_1
\end{aligned}
\qquad
\begin{aligned}
\tau ::= \;& \alpha \\
\mid\;& \tau_1 \rightarrow \tau_2 \\
\sigma ::= \;& \tau \\
\mid\;& \forall \alpha.\ \sigma
\end{aligned}
$$

```
let cons :: ∀a. a → [a] → [a] = λx. λlst. x : lst in ..
```

```
let evil :: (∀a. a → a) → Unit = λid. id 42; id "foo"; in ..
```

```
(λid. .. (id 42) .. (id "foo") ..) (λx. x)
```

$$
\begin{aligned}
e :=\ & x \\
\mid\ & \lambda x.\ e \\
\mid\ & e_1\ e_2 \\
\mid\ & \textbf{let}\ x = e_2\ \textbf{in}\ e_1 \\
p :=\ & e \\
\mid\ & \textbf{inst}\ o : \sigma_T = e\ \textbf{in}\ p
\end{aligned}
$$

$$
\begin{aligned}
\tau :=\ & \alpha \\
\mid\ & \tau_1 \to \tau_2 \\
\pi_\alpha :=\ & o_i : \alpha \to \tau_i \qquad i \in \mathbb{N} \\
\sigma :=\ & \tau \\
\mid\ & \forall \alpha . \pi_\alpha \Rightarrow \sigma
\end{aligned}
$$

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

System O — Syntax

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

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

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

System O — Typing

$$\Gamma = \{\mathbf{eq} : \mathbb{N} \to \mathbb{N} \to \mathbb{B},$$

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

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{r}\mathbf{eq} : \forall \alpha.\ (\mathbf{eq} : \alpha \to \alpha \to \mathbb{B}) \\ \Rightarrow [\alpha] \to [\alpha] \to \mathbb{B} \in \Gamma\end{array}}{\begin{array}{r}\Gamma \vdash \mathbf{eq} : \forall \alpha.\ (\mathbf{eq} : \alpha \to \alpha \to \mathbb{B}) \\ \Rightarrow [\alpha] \to [\alpha] \to \mathbb{B}\end{array}} \qquad \dfrac{\mathbf{eq} : \mathbb{N} \to \mathbb{N} \to \mathbb{B} \in \Gamma}{\Gamma \vdash \mathbf{eq} : \mathbb{N} \to \mathbb{N} \to \mathbb{B}}}{\Gamma \vdash \mathbf{eq} : [\mathbb{N}] \to [\mathbb{N}] \to \mathbb{B}} \qquad \dfrac{\ldots}{\Gamma \vdash [0] : [\mathbb{N}]}}{\Gamma \vdash \mathbf{eq}\ [0] : [\mathbb{N}] \to \mathbb{B}} \qquad \dfrac{\ldots}{\Gamma \vdash [0] : [\mathbb{N}]}}{\Gamma \vdash \mathbf{eq}\ [0]\ [0] : \mathbb{B}}$$

System O — Constraint Solving

$$
\begin{aligned}
&[\![\mathbf{inst}\ eq : \mathbb{N} \to \mathbb{N} \to \mathbb{B} = e_1\ \mathbf{in}\\
&\quad \mathbf{inst}\ eq : \forall \alpha.\ (\mathrm{eq} : \alpha \to \alpha \to \mathbb{B}) \Rightarrow [\alpha] \to [\alpha] \to \mathbb{B} = e_2\ \mathbf{in}\\
&\quad eq\ [0]\ [0]]\!]_{\emptyset}\\
= &[\![\mathbf{inst}\ eq : \forall \alpha.\ (\mathrm{eq} : \alpha \to \alpha \to \mathbb{B}) \Rightarrow [\alpha] \to [\alpha] \to \mathbb{B} = ..\ \mathbf{in}\\
&\quad eq\ [0]\ [0]]\!]_{\{eq := \lambda x.\ \mathbf{if}\ x\ \mathbf{is}\ \mathbb{N}\ \mathbf{then}\ [\![e_1]\!]\ x\}}\\
= &[\![eq\ [0]\ [0]]\!]_{\{eq := \lambda x.\ \mathbf{if}\ x\ \mathbf{is}\ \mathbf{List}\ \mathbf{then}\ [\![e_2]\!]\ x\ \mathbf{else}\ \mathbf{if}\ x\ \mathbf{is}\ \mathbb{N}\ \mathbf{then}\ [\![e_1]\!]\ x\}}\\
= &(\lambda x.\ \mathbf{if}\ x\ \mathbf{is}\ \mathbf{List}\ \mathbf{then}\ [\![e_2]\!]\ x\ \mathbf{else}\ \mathbf{if}\ x\ \mathbf{is}\ \mathbb{N}\ \mathbf{then}\ [\![e_1]\!]\ x)\ [0]\ [0]\\
= &[\![e_2]\!]\ [0]\ [0]\\
= &\mathbf{true}
\end{aligned}
$$

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

System O

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

Hindley Milner

System O — 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 O

System O — Relationship with Record Typing

Repository

github.com/Mari-W/popl

References

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