A Second Look at Overloading

Marius Weidner — Chair of Programming Languages — Seminar '22

```haskell
class Eq a where
  eq :: a → a → Bool

instance Eq Nat where
  eq Zero    Zero    = True
  eq (Suc x) (Suc y) = eq x y
  eq _       _       = False

instance Eq a ⇒ Eq [a] where
  eq []       []       = True
  eq (x : xs) (y : ys) = eq x y &&
                         eq xs ys
  eq _        _        = False

isEq :: Bool
isEq = eq [Zero] [Zero]
```

$$
\begin{aligned}
e :=\ & x \\
\mid\ & \lambda x.\ e \\
\mid\ & e\ e \\
\mid\ & \textbf{let}\ x = e\ \textbf{in}\ e
\end{aligned}
\qquad\qquad
\begin{aligned}
\tau :=\ & \alpha \\
\mid\ & \tau \to \tau \\
\sigma :=\ & \tau \\
\mid\ & \forall \alpha.\ \sigma
\end{aligned}
$$

```
let id = λx. x : ∀α. α → α in foo (id "Answer: ") (id 42)
```

$$\text{(TAUT)} \quad \frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad \frac{\Gamma \vdash e' : \sigma \qquad \Gamma,\ x : \sigma \vdash e : \tau}{\Gamma \vdash \textbf{let } x = e' \textbf{ in } e : \tau} \quad \text{(LET)}$$

$$(\to \mathrm{I}) \quad \frac{\Gamma,\ x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x.\ e : \tau \to \tau'} \qquad \frac{\Gamma \vdash e : \tau \to \tau' \qquad \Gamma \vdash e' : \tau}{\Gamma \vdash e\ e' : \tau'} \quad (\to \mathrm{E})$$

$$(\forall \mathrm{I}) \quad \frac{\Gamma \vdash e : \sigma \qquad \text{fresh } \alpha}{\Gamma \vdash e : \forall \alpha.\ \sigma} \qquad \frac{\Gamma \vdash e : \sigma \qquad \sigma \sqsubseteq \sigma'}{\Gamma \vdash e : \sigma'} \quad (\forall \mathrm{E})$$

Hindley Milner —— Typing

Let $\Gamma = \{\mathrm{id} : \forall\alpha.\ \alpha \to \alpha,\ \mathrm{n} : \mathrm{Nat}\}$.

$$\dfrac{\dfrac{\mathrm{id} : \forall\alpha.\ \alpha \to \alpha \in \Gamma \qquad \forall\alpha.\ \alpha \to \alpha \sqsubseteq \mathrm{Nat} \to \mathrm{Nat}}{\Gamma \vdash \mathrm{id} : \mathrm{Nat} \to \mathrm{Nat}} \qquad \dfrac{\mathrm{n} : \mathrm{Nat} \in \Gamma}{\Gamma \vdash \mathrm{n} : \mathrm{Nat}}}{\Gamma \vdash \mathrm{id\ n}}$$

Hindley Milner — Instantiation

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \{x : \alpha\}}{x : \alpha \vdash x : \alpha}}{\vdash \lambda x.\ x : \alpha \to \alpha}}{\vdash \lambda x.\ x : \forall \alpha.\ \alpha \to \alpha \quad \text{fresh } \alpha} \qquad \dfrac{\mathrm{id} : \forall \alpha.\ \alpha \to \alpha \in \{\mathrm{id} : \forall \alpha.\ \alpha \to \alpha\}}{\mathrm{id} : \forall \alpha.\ \alpha \to \alpha \vdash \mathrm{id} : \forall \alpha.\ \alpha \to \alpha}}{\vdash \textbf{let } \mathrm{id} = \lambda x.\ x \textbf{ in } \mathrm{id} : \forall \alpha.\ \alpha \to \alpha}$$

Hindley Milner —— Generalization

$$
\begin{aligned}
e :=\ & x \\
\mid\ & o \quad (\text{ if overloaded}) \\
\mid\ & k \quad (k \in \{\mathrm{unit},\ 42,\ [e_1, .., e_n],\ ..\}) \\
\mid\ & \lambda x.\ e \\
\mid\ & e\ e \\
\mid\ & \mathbf{let}\ x = e\ \mathbf{in}\ e
\end{aligned}
$$

$$
\begin{aligned}
p :=\ & e \\
\mid\ & \mathbf{inst}\ o : \sigma_T = \sigma\ \mathbf{in}\ p
\end{aligned}
$$

```
inst eq : Nat → Nat → Bool = λx. λy. x ≐ y in
inst eq : ∀α. (eq : α → α → Bool) ⇒ [α] → [α] → Bool =
     | λ[]. λ[]. True
     | λ[x : xs].λ[y : ys]. eq x y && eq xs ys in
 eq [0] [0]
```

$$
\begin{aligned}
\tau &:= \alpha \\
&\mid \tau \to \tau \\
&\mid D\ \tau_1\ \ldots\ \tau_n \qquad (D \in \{\mathrm{Unit},\ \mathrm{Nat},\ \mathrm{List}\ \tau,\ ..\},\ \mathrm{arity}(D) = n) \\
\pi_\alpha &:= o_1 : \alpha \to \tau_1,\ \ldots\ , o_n : \alpha \to \tau_n \qquad (n \in \mathbb{N},\ o_i \neq o_j) \\
\sigma &:= \tau \\
&\mid \forall \alpha . \pi_\alpha \Rightarrow\ \sigma_T \\
\sigma_T &:= T\ \alpha_1\ \ldots\ \alpha_n \to \tau \qquad (T \in D \cup \{\to\},\ \mathrm{tv}(\tau) \subseteq \{\alpha_1, .., \alpha_n\}) \\
&\mid \forall \alpha . \pi_\alpha \Rightarrow\ \sigma_T \qquad (\mathrm{tv}(\pi_\alpha) \subseteq \mathrm{tv}(\sigma_T))
\end{aligned}
$$

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

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

$$(\mathrm{SET})\quad \frac{\Gamma \vdash x_1 : \sigma_1 \quad \ldots \quad \Gamma \vdash x_n : \sigma_n}{\Gamma \vdash x_1 : \sigma_1 \quad \ldots \quad x_n : \sigma_n}$$

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

$$\texttt{Let } \Gamma = \\ \mathrm{eq} : \mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Bool}, \\ \mathrm{eq} : \forall \alpha.\ (\mathrm{eq} : \alpha \to \alpha \to \mathrm{Bool}) \Rightarrow [\alpha] \to [\alpha] \to \mathrm{Bool}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathrm{eq} : \forall \alpha.\ (\mathrm{eq} : \alpha \to \alpha \to \mathrm{Bool}) \\ \Rightarrow [\alpha] \to [\alpha] \to \mathrm{Bool} \in \Gamma\end{array}}{\begin{array}{c}\Gamma \vdash \mathrm{eq} : \forall \alpha.\ (\mathrm{eq} : \alpha \to \alpha \to \mathrm{Bool}) \\ \Rightarrow [\alpha] \to [\alpha] \to \mathrm{Bool}\end{array}} \quad \dfrac{\mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Bool} \in \Gamma}{\Gamma \vdash \mathrm{Nat} \to \mathrm{Nat} \to \mathrm{Bool}}}{\Gamma \vdash \mathrm{eq} : [\mathrm{Nat}] \to [\mathrm{Nat}] \to \mathrm{Bool}} \quad \dfrac{\dots}{\Gamma \vdash [0] : \mathrm{Nat}}}{\Gamma \vdash \mathrm{eq}\ [0] : [\mathrm{Nat}] \to \mathrm{Bool}} \quad \dfrac{\dots}{\Gamma \vdash [0] : \mathrm{Nat}}}{\Gamma \vdash \mathrm{eq}\ [0]\ [0]}$$

System O — Constraint Solving

System O —— Compositional Semantics

```
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 eq_n→n→b = λ.. in
let eq_[α]→[α]→b = λeq_n→n→b. λ.. in
 eq_[α]→[α]→b eq_n→n→b [0] [0]
```

Hindley Milner

System O — Translation to Hindley Milner

Slides & Elaboration

github.com/Mari-W/popl

Literatur

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