A Second Look at Overloading

Marius Weidner — Lehrstuhl für Programmiersprachen — Seminar '22

```haskell
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]
```

<p class="subtitle">Haskell</p> </div>
<div>

```rust
trait Eq
  fn eq(&self, rhs: &Self) → Bool

impl Eq for Nat
  fn eq(&self, rhs: &Self) → Bool
    match (self, rhs) {
      (Zero, Zero)       ⇒ True,
      (Suc(x), Suc(y)) ⇒ x.eq(y),
      (_, _)             ⇒ False
```

```
inst eq :: Nat -> Nat -> Bool
  eq Zero    Zero    = True
  eq (Suc x) (Suc 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

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

<p class="subtitle">Pseudocode</p>

$$
\begin{aligned}
&\mid \lambda x.\, e \\
&\mid \textbf{let}\ x = e\ \textbf{in}\ e
\end{aligned}
$$

</div> <div>

$$
\begin{aligned}
\tau &:= \alpha \\
&\mid \tau \to \tau \\
\sigma &:= \tau \\
&\mid \forall \alpha.\, \sigma
\end{aligned}
$$

</div> </div>


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

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

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

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

<p class="subtitle">Hindley Milner —— Typing</p>

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

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

<p class="subtitle">Hindley Milner —— Instantiation</p>

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

<p class="subtitle">Hindley Milner —— Generalization</p>

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

```
inst eq : Nat → Nat → Bool = λx. λy. x ≐ y in
inst eq : ∀ɑ. (eq : ɑ → ɑ → Bool) ⇒ [ɑ] → [ɑ] → Bool =
    | λ[]. λ[]. True
    | λ[x : xs].λ[y : ys]. eq x y && eq xs ys
    | λ_.λ_. False
in eq [4, 2] [4, 2]
```

<p class="subtitle">System O — Expressions</p>

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

<p class="subtitle">System O — Types</p>

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

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

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

$$(\mathrm{INST})\quad \frac{\Gamma \vdash e : \sigma_T \quad \Gamma,\ o : \sigma_T \vdash p : \sigma \quad \forall(o : \sigma'_T) \in \Gamma \Rightarrow \sigma'_T \neq \sigma_T}{\Gamma \vdash \mathbf{inst}\ o : \sigma_T = e\ \mathbf{in}\ p : \sigma}$$

<p class="subtitle">System O -- Typing</p>

<p class="subtitle">System 0 —— Derivation</p>

<p class="subtitle">System 0 —— Semantics</p>

<p class="subtitle">System O —— Translation to Hindley Milner</p>

Folien & Code

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