Type Inference

- Instead of writing type annotations, can we use an algorithm to infer what the type annotations should be?
- That depends on the type system. For simple type systems the answer is yes, and sophisticated type systems the answer is often no (it's undecidable.)
- We'll first look at type inference for the simply typed lambda calculus.
- Then we'll look at type inference in ML (i.e., Hindley-Milner inference).

The Simply Typed Lambda Calculus

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\ e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \to \tau_3 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1\ e_2) : \tau_3}$$

The main constraint on the inferred types is that the type of e_2 needs to be equal to the parameter type of e_1.

An alternative presentation (from Milner) attaches types to every subexpression and expresses all the constraints as equalities.

$$\frac{x_\tau \in \Gamma}{\Gamma \vdash x_\tau} \qquad \frac{\Gamma, x_\rho \vdash e_\sigma \qquad \tau = \rho \to \sigma}{\Gamma \vdash (\lambda x_\rho.\ e_\sigma)_\tau}$$

$$\frac{\Gamma \vdash e_\rho \qquad \Gamma \vdash e'_\sigma \qquad \rho = \sigma \to \tau}{\Gamma \vdash (e_\rho\ e'_\sigma)_\tau}$$

Type Inference

- First initialize all the attached types with unique "unknowns", that is, type variables.
- For example:
  $((\lambda f.\ (f\ 1))\ (\lambda x.\ x))$ becomes
  $((\lambda f_{\alpha_1}.\ (f_{\alpha_2}\ 1_{\alpha_3})_{\alpha_4})_{\alpha_5}\ (\lambda x_{\alpha_6}.\ x_{\alpha_7})_{\alpha_8})_{\alpha_9}$
- We can change the type system to generate a set of equalities over these variables.

$$\frac{x_\alpha \in \Gamma}{\Gamma \vdash x_\beta \Rightarrow \{\alpha = \beta\}} \qquad \frac{\Gamma, x_\alpha \vdash e_\beta \Rightarrow C}{\Gamma \vdash (\lambda x_\alpha.\ e_\beta)_\gamma \Rightarrow \{\gamma = \alpha \to \beta\} \cup C}$$

$$\frac{\Gamma \vdash e_\alpha \Rightarrow C_1 \qquad \Gamma \vdash e'_\beta \Rightarrow C_2}{\Gamma \vdash (e_\alpha\ e'_\beta)_\gamma \Rightarrow \{\alpha = \beta \to \gamma\} \cup C_1 \cup C_2}$$

Example

For this program

$$((\lambda f_{\alpha_1}.\ (f_{\alpha_2}\ 1_{\alpha_3})_{\alpha_4})_{\alpha_5}\ (\lambda x_{\alpha_6}.\ x_{\alpha_7})_{\alpha_8})_{\alpha_9}$$

we generate the following equations:

$$\begin{aligned}
\alpha_3 &= \mathsf{int} \\
\alpha_1 &= \alpha_2 \\
\alpha_2 &= \alpha_3 \to \alpha_4 \\
\alpha_5 &= \alpha_1 \to \alpha_4 \\
\alpha_6 &= \alpha_7 \\
\alpha_8 &= \alpha_6 \to \alpha_7 \\
\alpha_5 &= \alpha_8 \to \alpha_9
\end{aligned}$$

Solving equations by unification

- We can solve the equations by a process of normalization that simplifies the set of equations.
- The two main rules are:
  - Reduction: replace an equation of the form $\tau_1 \to \tau_2 = \tau_3 \to \tau_4$ with two equations: $\tau_1 = \tau_3$ and $\tau_2 = \tau_4$.
  - Variable elimination: suppose there is a equation of the form $\alpha = \tau$. If $\alpha \in \mathrm{ftv}(\tau)$ then report an error. Otherwise substitute τ for α in all the other equations.
- The auxiliary rules are:
  - Replace $\tau = \alpha$ with $\alpha = \tau$.
  - Remove equations of the form $\alpha = \alpha$, $\mathsf{int} = \mathsf{int}$, $\mathsf{bool} = \mathsf{bool}$, etc.
  - If there is an equation $\tau = \tau'$ where the head type constructor differs on each side, then stop and report failure. (e.g., $\mathsf{int} = \mathsf{int} \to \mathsf{int}$)

Example

$$\boxed{\alpha_3 = \text{int}}, \alpha_1 = \alpha_2, \alpha_2 = \boxed{\alpha_3} \to \alpha_4, \alpha_5 = \alpha_1 \to \alpha_4, \alpha_6 = \alpha_7,$$
$$\alpha_8 = \alpha_6 \to \alpha_7, \alpha_5 = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \text{int}, \boxed{\alpha_1 = \alpha_2}, \alpha_2 = \text{int} \to \alpha_4, \alpha_5 = \boxed{\alpha_1} \to \alpha_4, \alpha_6 = \alpha_7,$$
$$\alpha_8 = \alpha_6 \to \alpha_7, \alpha_5 = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \text{int}, \alpha_1 = \alpha_2, \boxed{\alpha_2 = \text{int} \to \alpha_4}, \alpha_5 = \alpha_2 \to \alpha_4, \alpha_6 = \alpha_7,$$
$$\alpha_8 = \alpha_6 \to \alpha_7, \alpha_5 = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \text{int}, \alpha_1 = \boxed{\alpha_2}, \boxed{\alpha_2 = \text{int} \to \alpha_4}, \alpha_5 = \boxed{\alpha_2} \to \alpha_4, \alpha_6 = \alpha_7,$$
$$\alpha_8 = \alpha_6 \to \alpha_7, \alpha_5 = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \text{int}, \alpha_1 = \text{int} \to \alpha_4, \alpha_2 = \text{int} \to \alpha_4, \alpha_5 = (\text{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_6 \to \alpha_7, \alpha_5 = \alpha_8 \to \alpha_9$$

Example, continued

$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \boxed{\alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4},$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_6 \to \alpha_7, \boxed{\alpha_5} = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\boxed{\alpha_6 = \alpha_7}, \alpha_8 = \boxed{\alpha_6} \to \alpha_7, (\mathsf{int} \to \alpha_4) \to \alpha_4 = \alpha_8 \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \boxed{\alpha_8 = \alpha_7 \to \alpha_7}, (\mathsf{int} \to \alpha_4) \to \alpha_4 = \boxed{\alpha_8} \to \alpha_9$$
$$\Longrightarrow \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_7 \to \alpha_7, \boxed{(\mathsf{int} \to \alpha_4) \to \alpha_4 = (\alpha_7 \to \alpha_7) \to \alpha_9}$$
$$\Longrightarrow \text{reduction} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_7 \to \alpha_7, (\mathsf{int} \to \alpha_4) = (\alpha_7 \to \alpha_7), \alpha_4 = \alpha_9$$

Example, continued

$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_7 \to \alpha_7, \boxed{(\mathsf{int} \to \alpha_4) = (\alpha_7 \to \alpha_7)}, \alpha_4 = \alpha_9$$
$$== \text{reduction} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \alpha_7, \alpha_8 = \alpha_7 \to \alpha_7, \boxed{\mathsf{int} = \alpha_7}, \alpha_4 = \alpha_7, \alpha_4 = \alpha_9$$
$$== \text{flip} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \alpha_4, \alpha_2 = \mathsf{int} \to \alpha_4, \alpha_5 = (\mathsf{int} \to \alpha_4) \to \alpha_4,$$
$$\alpha_6 = \boxed{\alpha_7}, \alpha_8 = \boxed{\alpha_7} \to \boxed{\alpha_7}, \boxed{\alpha_7 = \mathsf{int}}, \alpha_4 = \boxed{\alpha_7}, \alpha_4 = \alpha_9$$
$$== \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \boxed{\alpha_4}, \alpha_2 = \mathsf{int} \to \boxed{\alpha_4}, \alpha_5 = (\mathsf{int} \to \boxed{\alpha_4}) \to \boxed{\alpha_4},$$
$$\alpha_6 = \mathsf{int}, \alpha_8 = \mathsf{int} \to \mathsf{int}, \alpha_7 = \mathsf{int}, \boxed{\alpha_4 = \mathsf{int}}, \boxed{\alpha_4} = \alpha_9$$
$$== \text{variable elimination} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \mathsf{int}, \alpha_2 = \mathsf{int} \to \mathsf{int}, \alpha_5 = (\mathsf{int} \to \mathsf{int}) \to \mathsf{int},$$
$$\alpha_6 = \mathsf{int}, \alpha_8 = \mathsf{int} \to \mathsf{int}, \alpha_7 = \mathsf{int}, \alpha_4 = \mathsf{int}, \boxed{\mathsf{int} = \alpha_9}$$
$$== \text{flip} \Longrightarrow$$
$$\alpha_3 = \mathsf{int}, \alpha_1 = \mathsf{int} \to \mathsf{int}, \alpha_2 = \mathsf{int} \to \mathsf{int}, \alpha_5 = (\mathsf{int} \to \mathsf{int}) \to \mathsf{int},$$
$$\alpha_6 = \mathsf{int}, \alpha_8 = \mathsf{int} \to \mathsf{int}, \alpha_7 = \mathsf{int}, \alpha_4 = \mathsf{int}, \alpha_9 = \mathsf{int}$$

Example: the solution

So for this program

$$((\lambda f_{\alpha_1}.\ (f_{\alpha_2}\ 1_{\alpha_3})_{\alpha_4})_{\alpha_5}\ (\lambda x_{\alpha_6}.\ x_{\alpha_7})_{\alpha_8})_{\alpha_9}$$

the solution is:

$$\begin{aligned}
\alpha_1 &= \mathsf{int} \to \mathsf{int},\\
\alpha_2 &= \mathsf{int} \to \mathsf{int},\\
\alpha_3 &= \mathsf{int},\\
\alpha_4 &= \mathsf{int},\\
\alpha_5 &= (\mathsf{int} \to \mathsf{int}) \to \mathsf{int},\\
\alpha_6 &= \mathsf{int},\\
\alpha_8 &= \mathsf{int} \to \mathsf{int},\\
\alpha_7 &= \mathsf{int},\\
\alpha_9 &= \mathsf{int}
\end{aligned}$$

So we have

$$((\lambda f_{\mathsf{int}\to\mathsf{int}}.\ (f_{\mathsf{int}\to\mathsf{int}}\ 1_{\mathsf{int}})_{\mathsf{int}})_{(\mathsf{int}\to\mathsf{int})\to\mathsf{int}}\ (\lambda x_{\mathsf{int}}.\ x_{\mathsf{int}})_{\mathsf{int}\to\mathsf{int}})_{\mathsf{int}}$$

Most general unifier

- Sometimes the solution is underconstrained and there are many solutions.
- for example, $(\lambda x_{\alpha_1}.\, x_{\alpha_2})_{\alpha_3}$ gives rise to the following equations $\{\alpha_1 = \alpha_2, \alpha_3 = \alpha_1 \to \alpha_2\}$.
- Here's some solutions:
  1. $\{\alpha_1 = \mathsf{int}, \alpha_2 = \mathsf{int}, \alpha_3 = \mathsf{int} \to \mathsf{int}\}$
  2. $\{\alpha_1 = \mathsf{bool}, \alpha_2 = \mathsf{bool}, \alpha_3 = \mathsf{bool} \to \mathsf{bool}\}$
  3. $\{\alpha_1 = \alpha_2, \alpha_3 = \alpha_2 \to \alpha_2\}$
- Which solution is the best?
- The *most general unifier* is the solution that can be instantiated to match any other solution.
- Solution 3 is the most general unifier. Instantiate $\alpha_2 \mapsto \mathsf{int}$ to get solution 1 and $\alpha_2 \mapsto \mathsf{bool}$ to get solution 2.
- The unification algorithm always returns the most general unifier.

Hindley-Milner (ML-like) Inference

- The family of ML languages provide let-polymorphism, which is a restricted form of the parametric polymorphism in System F.
- Not only does the inference algorithm figure out the types, but it also figures out what should be generic and where instantiation should happen.
- Example:

  let f $= \lambda$ x. x **in**
  (f true, f 1)

  is equivalent to the following in System F

  let f $= \Lambda\alpha.\ \lambda$ x : α. x **in**
  (f[bool] true, f[int] 1)

The Hindley-Milner Type System

Universal quantification ($\forall$) is only allowed at the top of a type.

$$\begin{array}{rcl} T & ::= & \mathit{bool} \mid T \to T \\ S & ::= & T \mid \forall \overline{\alpha}.\ T \end{array}$$

The right-hand side of a "let" is inferred to be polymorphic.

$$\frac{\Gamma \vdash e_1 : T_1 \qquad \Gamma, x : \forall \overline{\alpha}.\ T_1 \vdash e_2 : T_2 \qquad \overline{\alpha} \cap \mathrm{FTV}(\Gamma) = \emptyset}{\Gamma \vdash \mathsf{let}\ x = e_1\ \mathsf{in}\ e_2 : T_2}$$

and instantiation happen implicitly when a variable is used

$$\frac{x : \forall \overline{\alpha}.\ T \in \Gamma}{\Gamma \vdash x : [\overline{\alpha} \mapsto \overline{T}]\, T}$$

Generalization

$$
\begin{array}{l}
\mathsf{generalize}(\Gamma,\ \tau) = \\
\quad \textbf{let}\ \overline{\alpha} = \mathsf{ftv}(\tau)\ \textbf{in} \\
\quad \textbf{let}\ \overline{\beta} = \mathsf{ftv}(\Gamma)\ \textbf{in} \\
\quad \textbf{let}\ \overline{\gamma} = \overline{\alpha} - \overline{\beta}\ \textbf{in} \\
\qquad \forall \overline{\gamma}.\ \tau
\end{array}
$$

Algorithm J

infer(Γ, e, E) =
  case e of
    $x \Rightarrow$
      let $\forall\overline{\alpha}.\ T = \Gamma(x)$ **and** $\overline{\beta}$ be fresh type variables **in**
        $(E, [\overline{\alpha} \mapsto \overline{\beta}]\, T)$
  | $(e_1\ e_2) \Rightarrow$
      let $(R, \rho) =$ infer(Γ, e_1, E) **in**
      let $(S, \sigma) =$ infer($R(\Gamma)$, e_2, R) **in**
      let $U =$ unify($\{S(\rho) = \sigma \rightarrow \beta\} \cup S$) where β is fresh **in**
        $(U, U(\beta))$
  | $\lambda x.\ e \Rightarrow$
      let $(R, \rho) =$ infer($(\Gamma, x : \beta)$, e, E) where β is fresh **in**
        $(R, R(\beta) \rightarrow \rho)$
  | let $x = e_1$ in $e_2 \Rightarrow$
      let $(R, \rho) =$ infer(Γ, e_1, E) **in**
      let $\tau =$ generalize($R(\Gamma)$, ρ) **in**
      let $(S, \sigma) =$ infer($(R(\Gamma), x : \tau)$, e_2, R) **in**
        (S, σ)

Algorithm W

infer(Γ, e) =
  case e of
    $x \Rightarrow$
      let $\forall\overline{\alpha}.\ T = \Gamma(x)$ **and** $\overline{\beta}$ be fresh type variables **in**
        $(\emptyset, [\overline{\alpha} \mapsto \overline{\beta}]\, T)$
  | $(e_1\ e_2) \Rightarrow$
      let $(R, \rho) =$ infer(Γ, e_1) **in**
      let $(S, \sigma) =$ infer($R(\Gamma)$, e_2) **in**
      let $U =$ unify($S(\rho) = \sigma \rightarrow \beta$) where β is fresh **in**
        $(U \circ S \circ R, U(\beta))$
  | $\lambda x.\ e \Rightarrow$
      let $(R, \rho) =$ infer($(\Gamma, x : \beta)$, e) where β is fresh **in**
        $(R, R(\beta) \rightarrow \rho)$
  | let $x = e_1$ in $e_2 \Rightarrow$
      let $(R, \rho) =$ infer(Γ, e_1) **in**
      let $\tau =$ generalize($R(\Gamma)$, ρ) **in**
      let $(S, \sigma) =$ infer($(R(\Gamma), x : \tau)$, e_2) **in**
        $(S \circ R, \sigma)$