Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC

https://courses.engr.illinois.edu/cs421/sp2023

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Two Problems

- Type checking
  - Question: Does exp. e have type τ in env Γ?
  - Answer: Yes / No
  - Method: Type derivation
- Typability
  - Question Does exp. e have some type in env. Γ? If so, what is it?
  - Answer: Type τ / error
  - Method: Type inference

Type Inference - Outline

- Begin by assigning a type variable as the type of the whole expression
- Decompose the expression into component expressions
- Use typing rules to generate constraints on components and whole
- Recursively find substitution that solves typing judgment of first subcomponent
- Apply substitution to next subcomponent and find substitution solving it; compose with first, etc.
- Apply comp of all substitution to orig. type var. to get answer

Type Inference - Example

- What type can we give to

  (fun x -> fun f -> f (f x))
- Start with a type variable and then look at the way the term is constructed

Type Inference - Example

- First approximate:

  { }|- (fun x -> fun f -> f (f x)) : α
- Second approximate: use fun rule

$$\frac{\{x : \beta\} \text{ |- (fun f -> f (f x))} : \gamma}{\{\ \} \text{ |- (fun x -> fun f -> f (f x))} : \alpha}$$

- Remember constraint $\alpha \equiv (\beta \rightarrow \gamma)$

Type Inference - Example

- Third approximate: use fun rule

$$\frac{\dfrac{\{f : \delta\ ;\ x : \beta\} \text{ |- f (f x)} : \varepsilon}{\{x : \beta\} \text{ |- (fun f -> f (f x))} : \gamma}}{\{\ \} \text{ |- (fun x -> fun f -> f(f x))} : \alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma)$; $\gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Fourth approximate: use app rule

$$\frac{\{f:\delta;\ x:\beta\} \vdash f : \varphi \to \varepsilon \quad \{f:\delta;\ x:\beta\} \vdash f\ x : \varphi}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Fifth approximate: use var rule, get constraint $\delta \equiv \varphi \to \varepsilon$, Solve with same
- Apply to next sub-proof

$$\frac{\{f:\delta;\ x:\beta\} \vdash f : \varphi \to \varepsilon \quad \{f:\delta;\ x:\beta\} \vdash f\ x : \varphi}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \to \varepsilon\}$

$$\frac{\dots \quad \{f:\varphi \to \varepsilon;\ x:\beta\} \vdash f\ x : \varphi}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \to \varepsilon\}$ Use App Rule

$$\frac{\frac{\{f:\varphi\to\varepsilon;\ x:\beta\} \vdash f:\zeta\to\varphi \quad \{f:\varphi\to\varepsilon;\ x:\beta\} \vdash x:\zeta}{\dots \quad \{f:\varphi \to \varepsilon;\ x:\beta\} \vdash f\ x : \varphi}}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \to \varepsilon\}$
- Var rule: Solve $\zeta\to\varphi \equiv \varphi\to\varepsilon$ Unification

$$\frac{\frac{\{f:\varphi\to\varepsilon;\ x:\beta\} \vdash f:\zeta\to\varphi \quad \{f:\varphi\to\varepsilon;\ x:\beta\} \vdash x:\zeta}{\dots \quad \{f:\varphi \to \varepsilon;\ x:\beta\} \vdash f\ x : \varphi}}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Current subst: $\{\zeta \equiv \varepsilon,\ \varphi \equiv \varepsilon\}$ o $\{\delta \equiv \varphi \to \varepsilon\}$
- Var rule: Solve $\zeta\to\varphi \equiv \varphi\to\varepsilon$ Unification

$$\frac{\frac{\{f:\varphi\to\varepsilon;\ x:\beta\} \vdash f:\zeta\to\varphi \quad \{f:\varphi\to\varepsilon;\ x:\beta\} \vdash x:\zeta}{\dots \quad \{f:\varphi \to \varepsilon;\ x:\beta\} \vdash f\ x : \varphi}}{\frac{\{f : \delta\ ;\ x : \beta\} \vdash (f\ (f\ x)) : \varepsilon}{\frac{\{x : \beta\} \vdash (\text{fun } f \to f\ (f\ x)) : \gamma}{\{\ \} \vdash (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}}}$$

- $\alpha \equiv (\beta \to \gamma);\ \gamma \equiv (\delta \to \varepsilon)$

Type Inference - Example

- Current subst: $\{\zeta\equiv\varepsilon,\ \varphi\equiv\varepsilon,\ \delta\equiv\varepsilon\rightarrow\varepsilon\}$
- Apply to next sub-proof

… {f:ε→ε; x:β}|- x:ε

… {f:φ → ε; x:β}|- f x : φ

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst: $\{\zeta\equiv\varepsilon,\ \varphi\equiv\varepsilon,\ \delta\equiv\varepsilon\rightarrow\varepsilon\}$
- Var rule: $\varepsilon\equiv\beta$

… {f:ε→ε; x:β}|- x:ε

… {f:φ → ε; x:β}|- f x : φ

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst: $\{\varepsilon\equiv\beta\}\text{o}\{\zeta\equiv\varepsilon,\ \varphi\equiv\varepsilon,\ \delta\equiv\varepsilon\rightarrow\varepsilon\}$
- Solves subproof; return one layer

… {f:ε→ε; x:β}|- x:ε

… {f:φ → ε; x:β}|- f x : φ

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst: $\{\varepsilon\equiv\beta,\ \zeta\equiv\beta,\ \varphi\equiv\beta,\ \delta\equiv\beta\rightarrow\beta\}$
- Solves this subproof; return one layer

…

… {f:φ → ε; x:β}|- f x : φ

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst: $\{\varepsilon\equiv\beta,\ \zeta\equiv\beta,\ \varphi\equiv\beta,\ \delta\equiv\beta\rightarrow\beta\}$
- Need to satisfy constraint $\gamma \equiv (\delta \rightarrow \varepsilon)$, given subst, becomes: $\gamma \equiv ((\beta\rightarrow\beta) \rightarrow \beta)$

…

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst:

$\{\gamma \equiv ((\beta\rightarrow\beta) \rightarrow \beta),\varepsilon\equiv\beta,\ \zeta\equiv\beta,\ \varphi\equiv\beta,\ \delta\equiv\beta\rightarrow\beta\}$

- Solves subproof; return one layer

…

{f : δ ; x : β} |- (f (f x)) : ε

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \rightarrow \gamma);\ \gamma \equiv (\delta \rightarrow \varepsilon)$

Type Inference - Example

- Current subst:

$\{\gamma \equiv ((\beta\to\beta) \to \beta), \varepsilon\equiv\beta, \zeta\equiv\beta, \varphi\equiv\beta, \delta\equiv\beta\to\beta\}$

- Need to satisfy constraint $\alpha \equiv (\beta \to \gamma)$ given subst: $\alpha \equiv (\beta \to ((\beta\to\beta) \to \beta))$

…

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

- $\alpha \equiv (\beta \to \gamma)$;

Type Inference - Example

- Current subst:

$\{\alpha \equiv (\beta \to ((\beta\to\beta) \to \beta)),$

$\gamma \equiv ((\beta\to\beta) \to \beta), \varepsilon\equiv\beta, \zeta\equiv\beta, \varphi\equiv\beta, \delta\equiv\beta\to\beta\}$

- Solves subproof; return one layer

{x : β} |- (fun f -> f (f x)) : γ

{ } |- (fun x -> fun f -> f (f x)) : α

Type Inference - Example

- Current subst:

$\{\alpha \equiv (\beta \to ((\beta\to\beta) \to \beta)),$

$\gamma \equiv ((\beta\to\beta) \to \beta), \varepsilon\equiv\beta, \zeta\equiv\beta, \varphi\equiv\beta, \delta\equiv\beta\to\beta\}$

- Done: $\alpha \equiv (\beta \to ((\beta\to\beta) \to \beta))$

{ } |- (fun x -> fun f -> f (f x)) : α

Type Inference Algorithm

Let infer (Γ, *e*, τ) = σ

- Γ is a typing environment (giving polymorphic types to expression variables)
- *e* is an expression
- τ is a type (with type variables),
- σ is a substitution of types for type variables
- Idea: σ is substitution solving the constraints on type variables necessary for Γ |- *e* : τ
- Should have $\sigma(\Gamma)$ |- *e* : $\sigma(\tau)$ valid

Type Inference Algorithm

infer (Γ, *exp*, τ) =

- Case *exp* of
  - Var *v* --> return Unify{$\tau \equiv$ freshInstance(Γ(*v*))}
    - Replace all quantified type vars by fresh ones
  - Const *c* --> return Unify{$\tau \equiv$ freshInstance φ } where Γ |- *c* : φ by the constant rules
  - fun *x* -> *e* -->
    - Let α, β be fresh variables
    - Let σ = infer ({x: α} + Γ, *e*, β)
    - Return Unify({$\sigma(\tau) \equiv \sigma(\alpha \to \beta)$}) $\circ$ σ

Example of inference with Var Rule

Instance {'a -> 'w}   ('w a fresh variable)

{x: All 'a. ('a * 'b) list, y:All. 'b}|- x : (int * string) list

freshInstance(All 'a. ('a * 'b) list) = ('w * 'b) list

Unify {((int*string)list = ('w * 'b) list)} ={'w -> int, 'b -> string}

After substitution:

Instance {'a -> int}

{x:All 'a. ('a * string) list, y:All. string}|- x:(int * string) list

Type Inference Algorithm (cont)

- Case *exp* of
  - App (e_1 e_2) -->
    - Let α be a fresh variable
    - Let σ_1 = infer(Γ, e_1, $\alpha \rightarrow \tau$)
    - Let σ_2 = infer($\sigma_1(\Gamma)$, e_2, $\sigma_1(\alpha)$)
    - Return $\sigma_2 \circ \sigma_1$

Type Inference Algorithm (cont)

- Case *exp* of
  - If e_1 then e_2 else e_3 -->
    - Let σ_1 = infer(Γ, e_1, bool)
    - Let σ_2 = infer($\sigma_1\Gamma$, e_2, $\sigma_1(\tau)$)
    - Let σ_3 = infer($\sigma_2 \circ \sigma_1(\Gamma)$,$e_3$,$\sigma_2 \circ \sigma_1(\tau)$)
    - Return $\sigma_3 \circ \sigma_2 \circ \sigma_1$

Type Inference Algorithm (cont)

- Case *exp* of
  - let $x = e_1$ in e_2 -->
    - Let α be a fresh variable
    - Let σ_1 = infer(Γ, e_1, α)
    - Let σ_2 =
    infer({x:GEN($\sigma_1(\Gamma)$, $\sigma_1(\alpha)$)} + $\sigma_1(\Gamma)$, e_2, $\sigma_1(\tau)$)
    - Return $\sigma_2 \circ \sigma_1$

Type Inference Algorithm (cont)

- Case *exp* of
  - let rec $x = e_1$ in e_2 -->
    - Let α be a fresh variable
    - Let σ_1 = infer({x: α} + Γ, e_1, α)
    - Let σ_2 = infer({x:GEN($\sigma_1(\Gamma)$,$\sigma_1(\alpha)$)}
    + $\sigma_1(\Gamma)$}, e_2, $\sigma_1(\tau)$)
    - Return $\sigma_2 \circ \sigma_1$

Type Inference Algorithm (cont)

- To infer a type, introduce type_of
- Let α be a fresh variable
- type_of (Γ, e) =
  - Let σ = infer (Γ, e, α)
  - Return σ (α)
- Need an algorithm for Unif