Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://www.cs.illinois.edu/class/cs421/

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 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

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

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

Type Inference - Example

- Fourth approximate: use app rule

$$\dfrac{\dfrac{\dfrac{[f:\delta;\ x:\beta]|\text{- } f : \varphi \rightarrow \varepsilon \quad [f:\delta;\ x:\beta]|\text{- } f\ x : \varphi}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dfrac{[f:\delta;\ x:\beta]|\text{- } f : \varphi \rightarrow \varepsilon \quad [f:\delta;\ x:\beta]|\text{- } f\ x : \varphi}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dfrac{\ldots \quad [f:\varphi \rightarrow \varepsilon;\ x:\beta]|\text{- } f\ x : \varphi}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dfrac{\ldots \quad \dfrac{[f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } f:\zeta\rightarrow\varphi \quad [f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } x:\zeta}{[f:\varphi \rightarrow \varepsilon;\ x:\beta]|\text{- } f\ x : \varphi}}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dfrac{\ldots \quad \dfrac{\dfrac{}{[f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } f:\zeta\rightarrow\varphi} \quad [f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } x:\zeta}{[f:\varphi \rightarrow \varepsilon;\ x:\beta]|\text{- } f\ x : \varphi}}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dfrac{\ldots \quad \dfrac{\dfrac{}{[f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } f:\zeta\rightarrow\varphi} \quad [f:\varphi\rightarrow\varepsilon;\ x:\beta]|\text{- } x:\zeta}{[f:\varphi \rightarrow \varepsilon;\ x:\beta]|\text{- } f\ x : \varphi}}{[f : \delta\ ;\ x : \beta]\ |\text{- } (f\ (f\ x)) : \varepsilon}}{[x : \beta]\ |\text{- } (\text{fun } f \text{ -> } f\ (f\ x)) : \gamma}}{[\ ]\ |\text{- } (\text{fun } x \text{ -> fun } f \text{ -> } f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

… $[f:\varepsilon \to \varepsilon; x:\beta]$ |- $x:\varepsilon$

… $[f:\varphi \to \varepsilon; x:\beta]$ |- $f\ x : \varphi$

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

Type Inference - Example

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

… $[f:\varepsilon \to \varepsilon; x:\beta]$ |- $x:\varepsilon$

… $[f:\varphi \to \varepsilon; x:\beta]$ |- $f\ x : \varphi$

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

Type Inference - Example

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

… $[f:\varepsilon \to \varepsilon; x:\beta]$ |- $x:\varepsilon$

… $[f:\varphi \to \varepsilon; x:\beta]$ |- $f\ x : \varphi$

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

Type Inference - Example

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

… $[f:\varphi \to \varepsilon; x:\beta]$ |- $f\ x : \varphi$

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

Type Inference - Example

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

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

Type Inference - Example

- Current subst:

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

- Solves subproof; return one layer

$[f : \delta ; x : \beta]$ |- $(f\ (f\ x)) : \varepsilon$

$[x : \beta]$ |- (fun f -> f (f x)) : γ

[ ] |- (fun x -> fun f -> f (f x)) : α

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

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

$$\frac{\overline{[x : \beta] \mid\!- (\text{fun f -> f (f x)}) : \gamma}}{[\ ] \mid\!- (\text{fun x -> fun f -> f (f x)}) : \alpha}$$

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

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 on layer

$$\frac{\overline{[x : \beta] \mid\!- (\text{fun f -> f (f x)}) : \gamma}}{[\ ] \mid\!- (\text{fun x -> fun f -> f (f x)}) : \alpha}$$

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

$[\ ] \mid\!- (\text{fun x -> fun f -> f (f x)}) : \alpha$

Type Inference Algorithm

Let infer $(\Gamma, e, \tau) = \sigma$

- 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 the constraints on type variables necessary for $\Gamma \mid\!- e : \tau$
- Should have $\sigma(\Gamma) \mid\!- e : \sigma(\tau)$

Type Inference Algorithm

has_type (Γ, *exp*, τ) =

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

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, $\alpha \to \tau$)
    - Let σ_2 = infer($\sigma(\Gamma)$, e_2, $\sigma(\alpha)$)
    - Return σ_2 o σ_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($\sigma\Gamma$, e_2, $\sigma_1(\tau)$)
    - Let σ_3 = infer($\sigma_2 \circ \sigma_1(\Gamma)$,$e_2$,$\sigma_2 \circ \sigma(\tau)$)
    - 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:[x:GEN($\sigma_1(\Gamma)$,$\sigma_1(\alpha)$)] + $\sigma_1(\Gamma)$] + $\sigma_1(\Gamma)$, 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 $\sigma(\alpha)$
- Need an algorithm for Unif