Programming Languages and Compilers (CS 421)

Reza Zamani

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 gather additional constraints to guarantee a solution for components
- Solve system of constraints to generate a substitution
- Apply 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:

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

- Second approximate: use fun rule

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

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

Type Inference - Example

- Third approximate: use fun rule

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

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

Type Inference - Example

- Fourth approximate: use app rule

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

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

Type Inference - Example

- Fifth approximate: use var rule

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

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

Type Inference - Example

- Sixth approximate: use var rule

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

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

Type Inference - Example

- Done building proof tree; now solve!

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

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

Type Inference - Example

- Type unification; solve like linear equations:

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

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

Type Inference - Example

- Eliminate φ:

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

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

Type Inference - Example

- Next eliminate δ :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{[f : \beta \to \varepsilon;\ x : \beta]\ |\!-\ f : \beta \to \varepsilon} \quad \dfrac{}{[f : \beta \to \varepsilon;\ x : \beta]\ |\!-\ x : \beta}}{[f : \beta \to \varepsilon;\ x : \beta]\ |\!-\ (f\ x) : \varepsilon}}{[x : \beta]\ |\!-\ (\text{fun } f \to f\ x) : \gamma}}{[\ ]\ |\!-\ (\text{fun } x \to \text{fun } f \to f\ x) : \alpha}$$

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

Type Inference - Example

- Next eliminate γ :

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

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

Type Inference - Example

- Next eliminate α :

$$\dfrac{\dfrac{\dfrac{\overline{[f : \beta \rightarrow \varepsilon;\ x : \beta] \mid\!- f : \beta \rightarrow \varepsilon} \quad \overline{[f : \beta \rightarrow \varepsilon;\ x : \beta] \mid\!- x : \beta}}{[f : \beta \rightarrow \varepsilon;\ x : \beta] \mid\!- (f\ x) : \varepsilon}}{[x : \beta] \mid\!- (\text{fun } f \text{ -> } f\ x) : ((\beta \rightarrow \varepsilon) \rightarrow \varepsilon)}}{[\,] \mid\!- (\text{fun } x \text{ -> fun } f \text{ -> } f\ x) : (\beta \rightarrow ((\beta \rightarrow \varepsilon) \rightarrow \varepsilon))}$$

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

Type Inference - Example

- No more equations to solve; we are done

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

- Any instance of $(\beta \rightarrow ((\beta \rightarrow \varepsilon) \rightarrow \varepsilon))$ is a valid type

Type Inference Algorithm

Let has_type (Γ, *e*, τ) = S

- Γ is a typing environment
- *e* is an expression
- τ is a (generalized) type,
- S is a set of equations between generalized types
- Idea: S is the constraints on type variables necessary for Γ |- *e* : τ
- Let Unif(S) be a substitution of generalized types for type variables solving S
- Solution: Unif(S)(Γ) |- *e* : Unif(S)(τ)

Type Inference Algorithm

has_type (Γ, *exp*, τ) =

- Case *exp* of
  - Var *v* --> return $\{\tau \equiv \Gamma(v)\}$
  - Const *c* --> return $\{\tau \equiv \sigma\}$ where Γ |- *c* : σ by the constant rules
  - fun *x* -> *e* -->
    - Let α, β be fresh variables
    - Let S = has_type ([x: α] + Γ, *e*, β)
    - Return $\{\tau \equiv \alpha \rightarrow \beta\} \cup$ S

Type Inference Algorithm (cont)

- Case *exp* of
  - App (e_1 e_2) -->
    - Let α be a fresh variable
    - Let S_1 = has_type(Γ, e_1, $\alpha \rightarrow \tau$)
    - Let S_2 = has_type(Γ, e_2, α)
    - Return $S_1 \cup S_2$

Type Inference Algorithm (cont)

- Case *exp* of
  - If e_1 then e_2 else e_3 -->
    - Let S_1 = has_type(Γ, e_1, bool)
    - Let S_2 = has_type(Γ, e_2, τ)
    - Let S_2 = has_type(Γ, e_2, τ)
    - Return $S_1 \cup S_2 \cup S_3$

Type Inference Algorithm (cont)

- Case *exp* of
  - let x = e_1 in e_2 -->
    - Let α be a fresh variable
    - Let S_1 = has_type(Γ, e_1, α)
    - Let S_2 =
      has_type([x: α] + Γ, e_2, τ)
    - Return $S_1 \cup S_2$

Type Inference Algorithm (cont)

- Case *exp* of
  - let rec $x = e_1$ in e_2 -->
    - Let α be a fresh variable
    - Let S_1 = has_type([x: α] + Γ, e_1, α)
    - Let S_2 = has_type([x: α] + Γ, e_2, τ)
    - Return $S_1 \cup S_2$

Type Inference Algorithm (cont)

- To infer a type, introduce type_of
- Let α be a fresh variable
- type_of (Γ, *e*) =
  - Let S = has_type (Γ, *e*, α)
  - Return Unif(S)(α)

- Need an algorithm for Unif