Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://www.cs.uiuc.edu/class/sp07/cs421/

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

Type Inference - The Problem

- Given an expression e, and a typing environment Γ, does there exist a type τ such that the judgment

  Γ |- e : τ

  is valid - ie., follows from the typing rules?

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:

  [ ] |- (fun x -> fun f -> f x) : α

- Second approximate: use fun rule

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

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

Type Inference - Example

- Third approximate: use fun rule

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

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

Type Inference - Example

- Fourth approximate: use app rule

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

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

Type Inference - Example

- Fifth approximate: use var rule

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

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

Type Inference - Example

- Sixth approximate: use var rule

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

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

Type Inference - Example

- Done building proof tree; now solve!

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

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

Type Inference - Example

- Type unification; solve like linear equations;

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

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

Type Inference - Example

- Eliminate φ:

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

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

Type Inference - Example

- Next eliminate δ :

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

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

Type Inference - Example

- Next eliminate γ :

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

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

Type Inference - Example

- Next eliminate α :

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

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

Type Inference - Example

- No more equations to solve; we are done

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

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

Type Inference - The Problem

- Given an expression *e*, and a typing environment Γ, does there exist a type τ such that the judgment

$$\Gamma \mid\!- e : \tau$$

is valid - ie., follows from the typing rules?

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

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] $\cup$ 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
  - 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] $\cup$ 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] $\cup$ Γ, e_1, α)
    - Let S_2 = has_type([x: α] $\cup$ 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