Programming Languages and Compilers (CS 421)

Elsa L Gunter

2112 SC, UIUC

http://courses.engr.illinois.edu/cs421

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

Review: In Class Activity

ACT 4

Mea Culpa

- The above system can't handle polymorphism as in OCAML
- No type variables in type language (only meta-variable in the logic)
- Would need:
  - Object level type variables and some kind of type quantification
  - **let** and **let rec** rules to introduce polymorphism
  - Explicit rule to eliminate (instantiate) polymorphism

Support for Polymorphic Types

- Monomorpic Types (τ):
  - Basic Types: int, bool, float, string, unit, …
  - Type Variables: α, β, γ, δ, ε
  - Compound Types: $\alpha \rightarrow \beta$, int * string, bool list, …
- Polymorphic Types:
  - Monomorphic types τ
  - Universally quantified monomorphic types
  - $\forall \alpha_1, \ldots, \alpha_n . \tau$
  - Can think of τ as same as $\forall . \tau$

Example FreeVars Calculations

- Vars('a -> (int -> 'b) -> 'a) ={'a , 'b}
- FreeVars (All 'b. 'a -> (int -> 'b) -> 'a) =
- {'a , 'b} – {'b}= {'a}
- FreeVars {
- x : All 'b. 'a -> (int -> 'b) -> 'a,
- id: All 'c. 'c -> 'c,
- y: All 'c. 'a -> ('b -> 'c)
- } = {'a} U {} U {'a, 'b} = {'a, 'b}

Support for Polymorphic Types

- Typing Environment Γ supplies polymorphic types (which will often just be monomorphic) for variables
- Free variables of monomorphic type just type variables that occur in it
  - Write FreeVars(τ)
- Free variables of polymorphic type removes variables that are universally quantified
  - $\text{FreeVars}(\forall\alpha_1, \ldots, \alpha_n . \tau) = \text{FreeVars}(\tau) - \{\alpha_1, \ldots, \alpha_n\}$
- FreeVars(Γ) = all FreeVars of types in range of Γ

Monomorphic to Polymorphic

- Given:
  - type environment Γ
  - monomorphic type τ
  - τ shares type variables with Γ
- Want most polymorphic type for τ that doesn't break sharing type variables with Γ
- $Gen(\tau, \Gamma) = \forall \alpha_1, \ldots , \alpha_n . \tau$ where $\{\alpha_1, \ldots , \alpha_n\} = freeVars(\tau) - freeVars(\Gamma)$

Polymorphic Typing Rules

- A *type judgement* has the form

$$\Gamma \text{ |- exp} : \tau$$

  - Γ uses polymorphic types
  - τ still monomorphic
- Most rules stay same (except use more general typing environments)
- Rules that change:
  - Variables
  - Let and Let Rec
  - Allow polymorphic constants
- Worth noting functions again

Polymorphic Let and Let Rec

- let rule:

$$\frac{\Gamma \mid\!- \; e_1 : \tau_1 \quad \{x : \mathrm{Gen}(\tau_1, \Gamma)\} + \Gamma \mid\!- \; e_2 : \tau_2}{\Gamma \mid\!- \; (\mathrm{let}\ x = e_1\ \mathrm{in}\ e_2) : \tau_2}$$

- let rec rule:

$$\frac{\{x : \tau_1\} + \Gamma \mid\!- \; e_1 : \tau_1 \quad \{x : \mathrm{Gen}(\tau_1, \Gamma)\} + \Gamma \mid\!- \; e_2 : \tau_2}{\Gamma \mid\!- \; (\mathrm{let\ rec}\ x = e_1\ \mathrm{in}\ e_2) : \tau_2}$$

Polymorphic Variables (Identifiers)

Variable axiom:

$$\frac{}{\Gamma \mid\!\!- x : \varphi(\tau)} \quad \text{if } \Gamma(x) = \forall \alpha_1, \ldots, \alpha_n \,.\, \tau$$

- Where φ replaces all occurrences of $\alpha_1, \ldots, \alpha_n$ by monotypes $\tau_1, \ldots, \tau_n$
- Note: Monomorphic rule special case:

$$\frac{}{\Gamma \mid\!\!- x : \tau} \quad \text{if } \Gamma(x) = \tau$$

- Constants treated same way

Fun Rule Stays the Same

- fun rule:

$$\frac{\{x : \tau_1\} + \Gamma \mid\!- \; e : \tau_2}{\Gamma \mid\!- \; \text{fun } x \text{ -> } e : \tau_1 \rightarrow \tau_2}$$

- Types τ_1, τ_2 monomorphic
- Function argument must always be used at same type in function body

Polymorphic Example

- Assume additional constants and primitive operators:
- hd : $\forall\alpha.\ \alpha$ list -> α
- tl: $\forall\alpha.\ \alpha$ list -> α list
- is_empty : $\forall\alpha.\ \alpha$ list -> bool
- (::) : $\forall\alpha.\ \alpha$ -> α list -> α list
- [] : $\forall\alpha.\ \alpha$ list

Polymorphic Example

- Show:

?

```
{} |- let rec length =
        fun l -> if is_empty l then 0
                 else 1 + length (tl l)
  in length (2 :: []) + length(true :: []) : int
```

Polymorphic Example: Let Rec Rule

- Show: (1) (2)

$$\frac{\{length{:}\alpha \text{ list} \to \text{int}\} \vdash \text{fun } l \to \ldots : \alpha \text{ list} \to \text{int} \qquad \{length{:}\forall\alpha.\ \alpha \text{ list} \to \text{int}\} \vdash \text{length } (2 :: []) + \text{length}(\text{true} :: []) : \text{int}}{\{\} \vdash \text{let rec length} = \text{fun } l \to \text{if is\_empty } l \text{ then } 0 \text{ else } 1 + \text{length } (\text{tl } l) \text{ in length } (2 :: []) + \text{length}(\text{true} :: []) : \text{int}}$$

Polymorphic Example (1)

- Show:

$$\frac{?}{\{length:\alpha\ list \to int\} \ |\text{-}\ \ fun\ l \to if\ is\_empty\ l\ then\ 0\ \ else\ 1 + length\ (tl\ l)\ :\ \alpha\ list \to int}$$

Polymorphic Example (1): Fun Rule

- Show: (3)

{length:α list -> int, l: α list } |-

if is_empty l then 0

else length (hd l) + length (tl l) : int

{length:α list -> int} |-

fun l -> if is_empty l then 0

else 1 + length (tl l)

: α list -> int

Polymorphic Example (3)

- Let Γ ={length:α list -> int, l: α list }
- Show

$$\frac{?}{\Gamma \text{ |- if is\_empty l then 0 else 1 + length (tl l) : int}}$$

Polymorphic Example (3):IfThenElse

- Let Γ ={length:α list -> int, l: α list }
- Show

$$\dfrac{\dfrac{(4)}{\Gamma \mid\!- \text{ is\_empty l : bool}} \quad \dfrac{(5)}{\Gamma \mid\!- \text{ 0:int}} \quad \dfrac{(6)}{\Gamma \mid\!- \text{ 1 + length (tl l) : int}}}{\Gamma \mid\!- \text{ if is\_empty l then 0 else 1 + length (tl l) : int}}$$

Polymorphic Example (4)

- Let Γ ={length:α list -> int, l: α list }
- Show

$$\frac{?}{\Gamma \text{|- is\_empty l : bool}}$$

Polymorphic Example (4):Application

- Let Γ ={length:α list -> int, l: α list }
- Show

$$\dfrac{\dfrac{?}{\Gamma |\text{-}\ \text{is\_empty} : \alpha\ \text{list} \rightarrow \text{bool}} \qquad \dfrac{?}{\Gamma |\text{-}\ \text{l} : \alpha\ \text{list}}}{\Gamma |\text{-}\ \text{is\_empty l} : \text{bool}}$$

Polymorphic Example (4)

- Let Γ ={length:α list -> int, l: α list }
- Show

By Const since α list -> bool
 is instance {$\alpha \rightarrow \alpha$} of
 $\forall \alpha$. α list -> bool ?

$$\dfrac{\dfrac{}{\Gamma \vdash \text{is\_empty} : \alpha \text{ list} \to \text{bool}} \qquad \dfrac{}{\Gamma \vdash \text{l} : \alpha \text{ list}}}{\Gamma \vdash \text{is\_empty l} : \text{bool}}$$

Polymorphic Example (4)

- Let Γ ={length:α list -> int, l: α list }
- Show

By Const since α list -> bool is instance of $\forall\alpha.\ \alpha$ list -> bool

By Variable Γ(l) = α list

$$\dfrac{\dfrac{}{\Gamma \mid\!- \text{ is\_empty} : \alpha \text{ list} \to \text{bool}} \qquad \dfrac{}{\Gamma \mid\!- \text{ l} : \alpha \text{ list}}}{\Gamma \mid\!- \text{ is\_empty l} : \text{bool}}$$

- This finishes (4)

Polymorphic Example (5):Const

- Let Γ ={length:α list -> int, l: α list }
- Show

By Const Rule

$$\frac{}{\Gamma |- 0:int}$$

Polymorphic Example (6): BinOp

- Let Γ ={length:α list -> int, l: α list }
- Show

$$
\dfrac{\dfrac{\text{By Const}}{\Gamma \mid\!- \; 1:\text{int}} \quad \dfrac{\dfrac{\text{By Variable}}{\Gamma \mid\!- \; \text{length} : \alpha \text{ list} \to \text{int}} \quad \dfrac{(7)}{\Gamma \mid\!- \; (\text{tl l}) : \alpha \text{ list}}}{\Gamma \mid\!- \; \text{length (tl l)} : \text{int}}\;\text{App}}{\Gamma \mid\!- \; 1 + \text{length (tl l)} : \text{int}}
$$

Polymorphic Example (7):App Rule

- Let Γ ={length:α list -> int, l: α list }
- Show

$$\dfrac{\dfrac{\text{Const}}{\Gamma \mid\text{- tl} : \alpha \text{ list} \to \alpha \text{ list}} \qquad \dfrac{\text{Variable}}{\Gamma \mid\text{- l} : \alpha \text{ list}}}{\Gamma \mid\text{- (tl l)} : \alpha \text{ list}}$$

By Const since α list -> α list is instance {$\alpha \rightarrow \alpha$} of $\forall\alpha$. α list -> α list

Polymorphic Example: (2) by BinOp

- Let Γ' = {length:$\forall\alpha.\ \alpha$ list -> int}
- Show:

$$\dfrac{\dfrac{(8)}{\Gamma' \text{ |- length (2 :: []) :int}} \qquad \dfrac{(9)}{\Gamma' \text{ |- length(true :: []) : int}}}{\text{\{length:}\forall\alpha.\ \alpha\text{ list -> int\} |- length (2 :: []) + length(true :: []) : int}}$$

Polymorphic Example: (8)AppRule

- Let Γ' = {length:$\forall\alpha$. α list -> int}
- Show:

$$\frac{\dfrac{?}{\Gamma' \text{ |- length : int list ->int}} \quad \dfrac{?}{\Gamma' \text{ |- (2 :: []):int list}}}{\Gamma' \text{ |- length (2 :: []) :int}}$$

Polymorphic Example: (8)AppRule

- Let Γ' = {length: $\forall\alpha.\ \alpha$ list -> int}
- Show:

$$\frac{\dfrac{?}{\Gamma' \text{ |- length : int list ->int}} \quad \dfrac{?}{\Gamma' \text{ |- (2 :: []):int list}}}{\Gamma' \text{ |- length (2 :: []) :int}}$$

Polymorphic Example: (8)AppRule

- Let Γ' = {length:∀α. α list -> int}
- Show:

By Var since int list -> int is instance {α → int} of ∀ α. α list -> int (by α ➔ int)

$$\dfrac{\dfrac{}{\Gamma\text{'} \;|\text{-}\; \text{length : int list -> int}} \qquad \dfrac{(10)}{\Gamma\text{'} \;|\text{-}\; \text{(2 :: []):int list}}}{\Gamma\text{'} \;|\text{-}\; \text{length (2 :: []) :int}}$$

Polymorphic Example: (10)BinOpRule

- Let Γ' = {length:$\forall\alpha. \alpha$ list -> int}
- Show:

$$\dfrac{\dfrac{\text{Const}}{\Gamma' \mid\!- 2 : \text{int}} \qquad \dfrac{?}{\Gamma' \mid\!- [\,] : \text{int list}}}{\Gamma' \mid\!- (2 :: [\,]) : \text{int list}}$$

Polymorphic Example: (10)BinOpRule

- Let Γ' = {length: $\forall\alpha.\ \alpha$ list -> int}
- Show:
- By Const since int list is instance of $\forall\alpha.\ \alpha$ list (by α ➔ int)

$$\dfrac{\dfrac{}{\Gamma' \text{ |- } 2 : \text{int}} \qquad \dfrac{}{\Gamma' \text{ |- } [] : \text{int list}}}{\Gamma' \text{ |- } (2 :: []) : \text{int list}}$$

Polymorphic Example: (9)AppRule

- Let Γ' = {length:$\forall\ \alpha.\ \alpha$ list -> int}
- Show:

$$\frac{\dfrac{?}{\Gamma' \text{ |- length :bool list ->int}} \qquad \dfrac{?}{\Gamma' \text{ |- (true :: []) :bool list}}}{\Gamma' \text{ |- length (true :: []) :int}}$$

Polymorphic Example: (9)AppRule

- Let Γ' = {length:$\forall$ α. α list -> int}
- Show:
- Var since bool list -> int is instance of $\forall\alpha$. α list -> int (by α ➔ bool)

$$\dfrac{\dfrac{}{\Gamma' \text{ |- length :bool list ->int}} \qquad \dfrac{(10)}{\Gamma' \text{ |- (true :: []) :bool list}}}{\Gamma' \text{ |- length (true :: []) :int}}$$

Polymorphic Example: (10)BinOpRule

- Let Γ' = {length:$\forall\alpha.\ \alpha$ list -> int}
- Show:

$$\dfrac{\dfrac{\text{Const}}{\Gamma' \text{ |- true : bool}} \qquad \dfrac{?}{\Gamma' \text{ |- [] : bool list}}}{\Gamma' \text{ |- (true :: []) : bool list}}$$

Polymorphic Example: (10)BinOpRule

- Let Γ' = {length: $\forall\alpha.\ \alpha$ list -> int}
- Show:
- By Const since bool list is instance of $\forall\alpha.\ \alpha$ list (by $\alpha \rightarrow$ bool)

$$\frac{\overline{\Gamma' \text{ |- true : bool}} \qquad \overline{\Gamma' \text{ |- [] : bool list}}}{\Gamma' \text{ |- (true :: []) : bool list}}$$

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:

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

- Second approximate: use fun rule

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

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

Type Inference - Example

- Third approximate: use fun rule

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

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

Type Inference - Example

- Fourth approximate: use app rule

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

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

Type Inference - Example

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

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

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

Type Inference - Example

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

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

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

Type Inference - Example

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

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

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

Type Inference - Example

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

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

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

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

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

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

Type Inference - Example

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

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

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

Type Inference - Example

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

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

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

Type Inference - Example

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

$$\dfrac{\dfrac{\dots \qquad \dfrac{}{\{f:\varepsilon\to\varepsilon;\ x:\beta\} \mid\!\!-\ x:\varepsilon}}{\dots \quad \{f:\varphi \to \varepsilon;\ x:\beta\} \mid\!\!-\ f\ x : \varphi}}{\{f : \delta\ ;\ x : \beta\} \mid\!\!-\ (f\ (f\ x)) : \varepsilon}$$

$$\dfrac{\dfrac{}{\{x : \beta\} \mid\!\!-\ (\text{fun } f \to f\ (f\ x)) : \gamma}}{\{\ \} \mid\!\!-\ (\text{fun } x \to \text{fun } f \to f\ (f\ x)) : \alpha}$$

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

Type Inference - Example

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

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

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

Type Inference - Example

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

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

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

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

$$\dfrac{\dfrac{\dfrac{\dots}{\{f : \delta \,;\, x : \beta\} \mid\!- (f\ (f\ x)) : \varepsilon}}{\{x : \beta\} \mid\!- (\text{fun } f \rightarrow f\ (f\ x)) : \gamma}}{\{\ \} \mid\!- (\text{fun } x \rightarrow \text{fun } f \rightarrow f\ (f\ x)) : \alpha}$$

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

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

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

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

Type Inference - Example

- Current subst:

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

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

- Solves subproof; return on layer

———————————————————

{x : β} |- (fun f -> f (f x)) : γ

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

Type Inference - Example

- Current subst:

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

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

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

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