

Programming Languages and Compilers (CS 421)

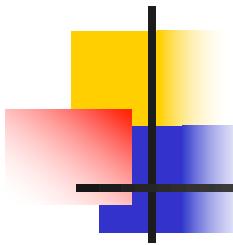


Elsa L Gunter

2112 SC, UIUC

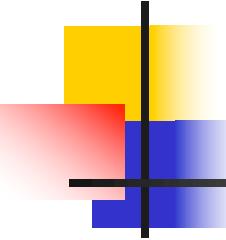
<http://courses.engr.illinois.edu/cs421>

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



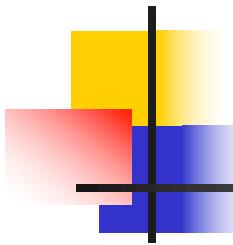
Review: In Class Activity

ACT 4



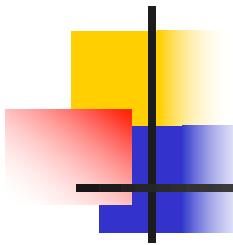
Mea Culpa

- 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



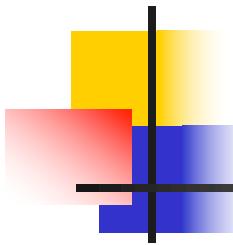
Support for Polymorphic Types

- Monomorphic Types (τ):
 - Basic Types: int, bool, float, string, unit, ...
 - Type Variables: $\alpha, \beta, \gamma, \delta, \varepsilon$
 - Compound Types: $\alpha \rightarrow \beta$, int * string, bool list, ...
- Polymorphic Types:
 - Monomorphic types τ
 - Universally quantified monomorphic types
 - $\forall \alpha_1, \dots, \alpha_n . \tau$
 - Can think of τ as same as $\forall . \tau$



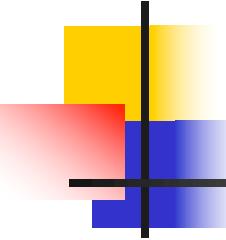
Example FreeVars Calculations

- $\text{Vars}(\text{'a} \rightarrow (\text{int} \rightarrow \text{'b}) \rightarrow \text{'a}) = \{\text{'a}, \text{'b}\}$
- $\text{FreeVars} (\text{All } \text{'b}. \text{'a} \rightarrow (\text{int} \rightarrow \text{'b}) \rightarrow \text{'a}) =$
- $\{\text{'a}, \text{'b}\} - \{\text{'b}\} = \{\text{'a}\}$
- $\text{FreeVars} \{$
- $x : \text{All } \text{'b}. \text{'a} \rightarrow (\text{int} \rightarrow \text{'b}) \rightarrow \text{'a},$
- $\text{id} : \text{All } \text{'c}. \text{'c} \rightarrow \text{'c},$
- $y : \text{All } \text{'c}. \text{'a} \rightarrow (\text{'b} \rightarrow \text{'c})$
- $\} = \{\text{'a}\} \cup \{\} \cup \{\text{'a}, \text{'b}\} = \{\text{'a}, \text{'b}\}$



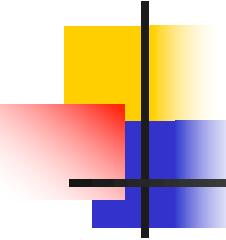
Support for Polymorphic Types

- 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 $\text{FreeVars}(\tau)$
- Free variables of polymorphic type removes variables that are universally quantified
 - $\text{FreeVars}(\forall \alpha_1, \dots, \alpha_n . \tau) = \text{FreeVars}(\tau) - \{\alpha_1, \dots, \alpha_n\}$
- $\text{FreeVars}(\Gamma) = \text{all } \text{FreeVars} \text{ of types in range of } \Gamma$



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
- $\text{Gen}(\tau, \Gamma) = \forall \alpha_1, \dots, \alpha_n . \tau$ where
 $\{\alpha_1, \dots, \alpha_n\} = \text{freeVars}(\tau) - \text{freeVars}(\Gamma)$

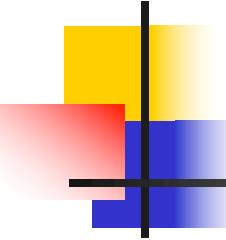


Polymorphic Typing Rules

- A *type judgement* has the form

$$\Gamma \vdash \text{exp} : \tau$$

- 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



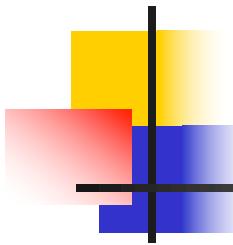
Polymorphic Let and Let Rec

- let rule:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \{x: \text{Gen}(\tau_1, \Gamma)\} + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{let } x = e_1 \text{ in } e_2) : \tau_2}$$

- let rec rule:

$$\frac{\{x: \tau_1\} + \Gamma \vdash e_1 : \tau_1 \quad \{x: \text{Gen}(\tau_1, \Gamma)\} + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{let rec } x = e_1 \text{ in } e_2) : \tau_2}$$



Polymorphic Variables (Identifiers)

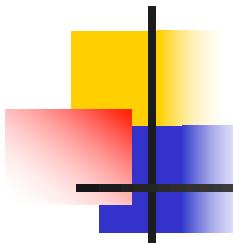
Variable axiom:

$$\frac{}{\Gamma \vdash x : \varphi(\tau)} \quad \text{if } \Gamma(x) = \forall \alpha_1, \dots, \alpha_n . \tau$$

- Where φ replaces all occurrences of $\alpha_1, \dots, \alpha_n$ by monotypes τ_1, \dots, τ_n
- Note: Monomorphic rule special case:

$$\frac{}{\Gamma \vdash x : \tau} \quad \text{if } \Gamma(x) = \tau$$

- Constants treated same way

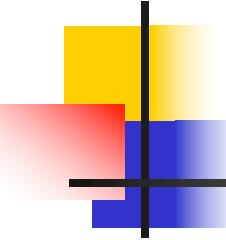


Fun Rule Stays the Same

- fun rule:

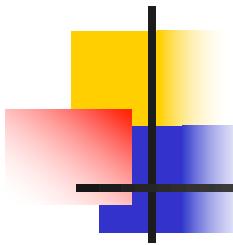
$$\frac{\{x: \tau_1\} + \Gamma \vdash e : \tau_2}{\Gamma \vdash \text{fun } x \rightarrow e : \tau_1 \rightarrow \tau_2}$$

- 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:
- $\text{hd} : \forall \alpha. \alpha \text{ list} \rightarrow \alpha$
- $\text{tl} : \forall \alpha. \alpha \text{ list} \rightarrow \alpha \text{ list}$
- $\text{is_empty} : \forall \alpha. \alpha \text{ list} \rightarrow \text{bool}$
- $(::) : \forall \alpha. \alpha \rightarrow \alpha \text{ list} \rightarrow \alpha \text{ list}$
- $[] : \forall \alpha. \alpha \text{ list}$

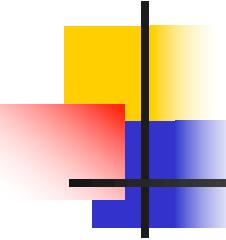


Polymorphic Example

- 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)

{length: α list -> int} {length: $\forall\alpha.$ α list -> int}

| - fun l -> ... | - length (2 :: []) +

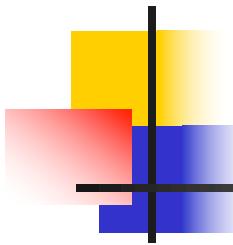
: α list -> int length(true :: []) : int

{ } | - let rec length =

fun l -> if is_empty l then 0

else 1 + length (tl l)

in length (2 :: []) + length(true :: []) : int

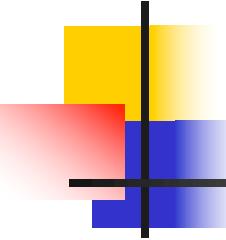


Polymorphic Example (1)

- Show:

?

```
{length: $\alpha$  list -> int} |-  
fun l -> if is_empty l then 0  
                      else 1 + length (tl l)  
:  $\alpha$  list -> int
```



Polymorphic Example (1): Fun Rule

- Show: (3)
-

$$\{ \text{length}: \alpha \text{ list} \rightarrow \text{int}, \quad l: \alpha \text{ list} \} \vdash$$

if `is_empty` l then 0

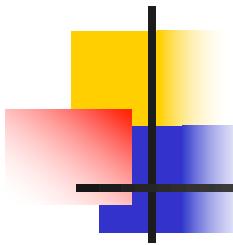
else `length` (`hd` l) + `length` (`tl` l) : int

$$\{ \text{length}: \alpha \text{ list} \rightarrow \text{int} \} \vdash$$

fun l -> if `is_empty` l then 0

else 1 + `length` (`tl` l)

: α list \rightarrow int

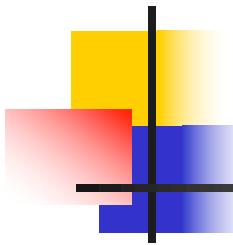


Polymorphic Example (3)

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

?

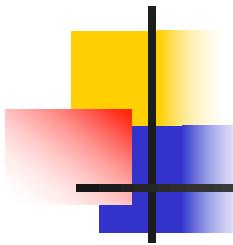
$$\begin{aligned} \Gamma |- & \text{ if } \text{is_empty } \text{l} \text{ then } 0 \\ & \text{else } 1 + \text{length } (\text{tl } \text{l}) : \text{int} \end{aligned}$$



Polymorphic Example (3): IfThenElse

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

$$\frac{(4) \quad \frac{}{\Gamma \vdash \text{is_empty } \text{l} : \text{bool}} \quad (5) \quad \frac{}{\Gamma \vdash 0 : \text{int}} \quad (6) \quad \frac{}{\Gamma \vdash 1 + \text{length} (\text{tl } \text{l}) : \text{int}}}{\Gamma \vdash \text{if is_empty } \text{l} \text{ then } 0 \text{ else } 1 + \text{length} (\text{tl } \text{l}) : \text{int}}$$

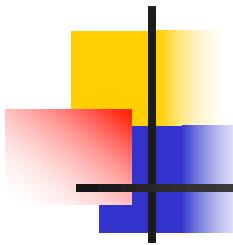


Polymorphic Example (4)

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

?

$\Gamma \vdash \text{is_empty } \text{l} : \text{bool}$



Polymorphic Example (4): Application

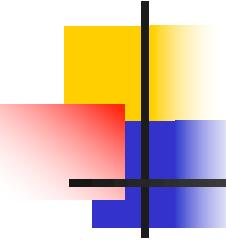
- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

?

?

$$\Gamma \vdash \text{is_empty} : \alpha \text{ list} \rightarrow \text{bool}$$

$$\Gamma \vdash \text{l} : \alpha \text{ list}$$
$$\Gamma \vdash \text{is_empty l} : \text{bool}$$



Polymorphic Example (4)

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

By Const since $\alpha \text{ list} \rightarrow \text{bool}$

is Instance: $\{\alpha \rightarrow \alpha\}$ of

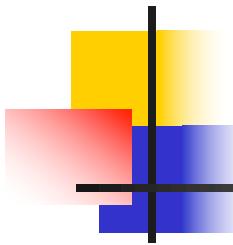
$\forall \alpha. \alpha \text{ list} \rightarrow \text{bool}$

?

$\Gamma \vdash \text{is_empty} : \alpha \text{ list} \rightarrow \text{bool}$

$\Gamma \vdash \text{l} : \alpha \text{ list}$

$\Gamma \vdash \text{is_empty l} : \text{bool}$



Polymorphic Example (4)

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

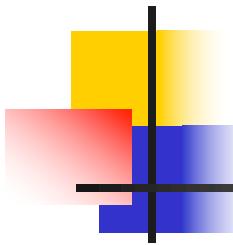
By Const since $\alpha \text{ list} \rightarrow \text{bool}$ is By Variable
instance of $\forall \alpha. \alpha \text{ list} \rightarrow \text{bool}$ $\Gamma(\text{l}) = \alpha \text{ list}$

$\Gamma \vdash \text{is_empty} : \alpha \text{ list} \rightarrow \text{bool}$

$\Gamma \vdash \text{l} : \alpha \text{ list}$

$\Gamma \vdash \text{is_empty l} : \text{bool}$

- This finishes (4)

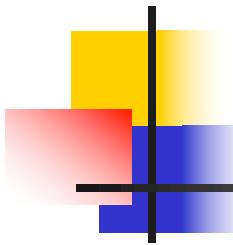


Polymorphic Example (5):Const

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list} \}$
- Show

By Const Rule

$$\frac{}{\Gamma \vdash 0 : \text{int}}$$



Polymorphic Example (6): BinOp

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

By Variable

$$\frac{}{\Gamma |- \text{length}} \quad (7)$$

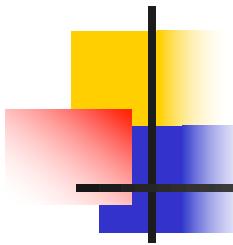
By Const

$$: \alpha \text{ list} \rightarrow \text{int} \quad \frac{}{\Gamma |- (\text{tl l}) : \alpha \text{ list}}$$

$$\frac{}{\Gamma |- 1 : \text{int}}$$

$$\text{App} \quad \frac{}{\Gamma |- \text{length} (\text{tl l}) : \text{int}}$$

$$\frac{}{\Gamma |- 1 + \text{length} (\text{tl l}) : \text{int}}$$

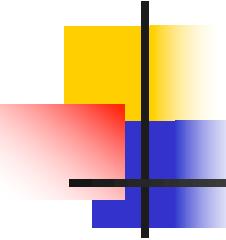


Polymorphic Example (7):App Rule

- Let $\Gamma = \{\text{length} : \alpha \text{ list} \rightarrow \text{int}, \text{ l} : \alpha \text{ list}\}$
- Show

| | |
|---|--|
| Const | Variable |
| <hr/> $\Gamma \vdash \text{tl} : \alpha \text{ list} \rightarrow \alpha \text{ list}$ | <hr/> $\Gamma \vdash \text{l} : \alpha \text{ list}$ |
| $\Gamma \vdash (\text{tl } \text{l}) : \alpha \text{ list}$ | |

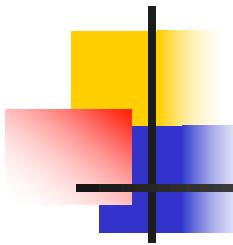
By Const since $\alpha \text{ list} \rightarrow \alpha \text{ list}$ is instance
 $\{\alpha \rightarrow \alpha\}$ of $\forall \alpha. \alpha \text{ list} \rightarrow \alpha \text{ list}$



Polymorphic Example: (2) by BinOp

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

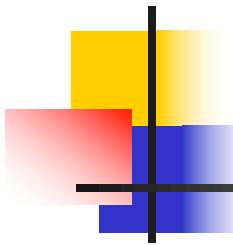
$$\frac{\begin{array}{c} (8) \\ \hline \Gamma' \vdash \text{length } (2 :: []) : \text{int} \end{array} \quad \begin{array}{c} (9) \\ \hline \Gamma' \vdash \text{length}(\text{true} :: []) : \text{int} \end{array}}{\begin{array}{c} \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\} \\ \vdash \text{length } (2 :: []) + \text{length}(\text{true} :: []) : \text{int} \end{array}}$$



Polymorphic Example: (8)AppRule

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

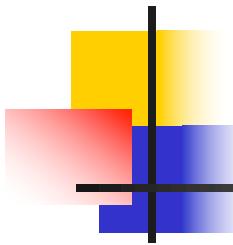
$$\frac{\begin{array}{c} ? \\ \hline \Gamma' \vdash \text{length} : \text{int list} \rightarrow \text{int} \end{array} \quad \begin{array}{c} ? \\ \hline \Gamma' \vdash (2 :: []) : \text{int list} \end{array}}{\Gamma' \vdash \text{length} (2 :: []) : \text{int}}$$



Polymorphic Example: (8)AppRule

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

$$\frac{\begin{array}{c} ? \\ \hline \Gamma' \vdash \text{length} : \text{int list} \rightarrow \text{int} \end{array} \quad \begin{array}{c} ? \\ \hline \Gamma' \vdash (2 :: []) : \text{int list} \end{array}}{\Gamma' \vdash \text{length} (2 :: []) : \text{int}}$$



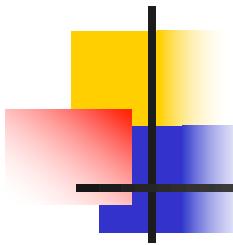
Polymorphic Example: (8)AppRule

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

By Var since $\text{int list} \rightarrow \text{int}$ is Instance: $\{\alpha \rightarrow \text{int}\}$
of $\forall \alpha. \alpha \text{ list} \rightarrow \text{int}$ (by $\alpha \rightarrow \text{int}$)

(10)

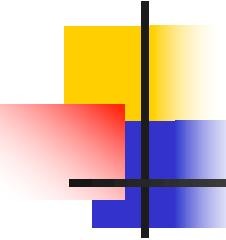
$$\frac{\Gamma' \vdash \text{length} : \text{int list} \rightarrow \text{int} \quad \Gamma' \vdash (2 :: []) : \text{int list}}{\Gamma' \vdash \text{length} (2 :: []) : \text{int}}$$



Polymorphic Example: (10)BinOpRule

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

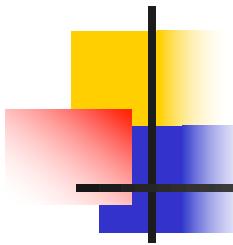
$$\frac{\begin{array}{c} \text{Const} \\ \hline \Gamma' \vdash 2 : \text{int} \end{array} \quad \frac{\text{?}}{\Gamma' \vdash [] : \text{int list}}}{\Gamma' \vdash (2 :: []) : \text{int list}}$$



Polymorphic Example: (10)BinOpRule

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

$$\frac{\overline{\Gamma' \vdash 2 : \text{int}} \quad \overline{\Gamma' \vdash [] : \text{int list}}}{\Gamma' \vdash (2 :: []) : \text{int list}}$$



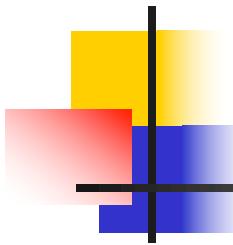
Polymorphic Example: (9)AppRule

- Let $\Gamma' = \{\text{length} \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

$$\frac{?}{\Gamma' \vdash \text{length} : \text{bool list} \rightarrow \text{int}}$$

$$\frac{?}{\Gamma' \vdash (\text{true} :: []) : \text{bool list}}$$

$$\Gamma' \vdash \text{length} (\text{true} :: []) : \text{int}$$



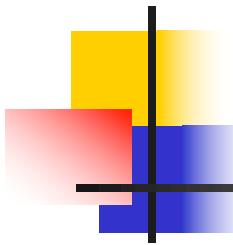
Polymorphic Example: (9)AppRule

- Let $\Gamma' = \{\text{length} \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:
- Var since $\text{bool list} \rightarrow \text{int}$ is instance of
 $\forall \alpha. \alpha \text{ list} \rightarrow \text{int}$ (by $\alpha \rightarrow \text{bool}$)

(10)

$$\frac{\Gamma' \vdash \text{length} : \text{bool list} \rightarrow \text{int}}{\Gamma' \vdash \text{length} (\text{true} :: []) : \text{int}}$$

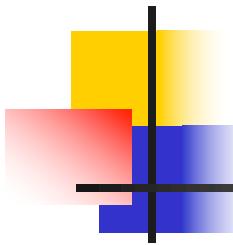
$$\frac{\Gamma' \vdash (\text{true} :: []) : \text{bool list}}{\Gamma' \vdash \text{length} (\text{true} :: []) : \text{int}}$$



Polymorphic Example: (10)BinOpRule

- Let $\Gamma' = \{\text{length} : \forall \alpha. \alpha \text{ list} \rightarrow \text{int}\}$
- Show:

$$\frac{\begin{array}{c} \text{Const} \\ \hline \Gamma' \vdash \text{true} : \text{bool} \end{array} \quad ? \quad \frac{\hline}{\Gamma' \vdash [] : \text{bool list}}}{\Gamma' \vdash (\text{true} :: []) : \text{bool list}}$$



Polymorphic Example: (10)BinOpRule

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

$$\frac{\Gamma' \vdash \text{true} : \text{bool} \quad \Gamma' \vdash [] : \text{bool list}}{\Gamma' \vdash (\text{true} :: []) : \text{bool list}}$$