# The Hindley-Milner Type System

Stephen A. Edwards

Columbia University

Spring 2023

## Simply-Typed Lambda Calculus

$$
\begin{array}{rl}
\text{Types} & \tau ::= \mathbf{bool} \mid \tau \longrightarrow \tau \\
\text{Values} & v ::= \mathbf{true} \mid \mathbf{false} \mid x \mid \lambda x : \tau \,.\, t \\
\text{Terms} & t ::= v \mid t\ t \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t
\end{array}
$$

Must annotate every function argument with a type

No polymorphism: expressions that don't get stuck are not well-typed, e.g., $(\lambda x : ?\ .\ x)\ y$

Principal type-schemes for functional programs

Luis Damas* and Robin Milner
Edinburgh University

1. Introduction

This paper is concerned with the polymorphic type discipline of ML, which is a general purpose functional programming language, although it was

of successful use of the language, both in LCF and other research and in teaching to undergraduates, it has become important to answer these questions - particularly because the combination of flexibility

Symposium on Principles of Programming Languages (POPL), 1982

Also R. Hindley, *The principal type-scheme of an object in Combinatory Logic*, Transactions of the American Mathematical Society, 146:29–60, December 1969.

So more like Damas-Hindley-Milner

## Hindley-Milner Type System

| | | | |
|---|---|---|---|
| $e$ | ::= | $x$ | Variable |
| | | $e\ e$ | Application |
| | | $\lambda x\ .\ e$ | Abstraction |
| | | **let** $x$ **=** $e$ **in** $e$ | Let binding |

The Lambda calculus extended with **let** bindings

**let** $x$ **=** $M$ **in** $N$ behaves like $(\lambda x\ .\ N)\ M$ but with different typing rules

## Hindley-Milner Type System

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | | |
| | | **let** $x$ **=** $e$ **in** $e$ | Let binding | | |

$\tau$: A type, which is *polymorphic* if it contains *type variables*; it is *monotype* otherwise

$\alpha$: A *type variable* that may be filled in later

$C$: A *type constructor* that takes zero or more type arguments (its *arity*)

E.g., **bool**, **int**, **char**, **string** $\in C$ with arity zero

**list** and **set** $\in C$ with arity one, e.g., **list int**

**map** and $\rightarrow \in C$ with arity two, e.g., **map string int**, **string** $\rightarrow$ **bool** (infix)

## Hindley-Milner Type System

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$\sigma$: A *type scheme* that can be *instantiated*

A *quantifier* $\forall\alpha\ .\ \sigma$ introduces a local type variable $\alpha$ to the scope of $\sigma$

Quantifiers may only occur at the top level, i.e., $\sigma = \forall\alpha_1\ .\ \forall\alpha_2\ .\ \ldots\ .\ \forall\alpha_n\ .\ \tau$

Type scheme of the identity function: $\lambda x\ .\ x : \forall\alpha\ .\ \alpha \longrightarrow \alpha$

Type scheme of a function that returns the length of a list: $\forall\alpha\ .\ (\textbf{list}\ \alpha) \longrightarrow \textbf{int}$

Type scheme of a function that returns the number of elements in a map:
$\forall\alpha_1\ .\ \forall\alpha_2\ .\ (\textbf{map}\ \alpha_1\ \alpha_2) \longrightarrow \textbf{int}$

## Hindley-Milner Type System

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

Types may include type variables $\alpha \longrightarrow \alpha$

Type schemes' type variables may be quantified $\forall\alpha_1\ .\ \alpha_1 \longrightarrow \alpha_2$ or unquantified $\alpha_1 \longrightarrow \alpha_2$

What is the difference?

*You choose* values for a quantified type, $\forall\alpha_1\ .\ \alpha_1 \longrightarrow \alpha_2$ may mean **int** $\longrightarrow \alpha_2$ or **bool** $\longrightarrow \alpha_2$.

An unquantified type variable may be *chosen by something else, but not you.*
$\alpha \longrightarrow \alpha$ means a function from some type $\alpha$ to that same type, but you may not choose $\alpha$.

## Hindley-Milner Type System

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

"$\sigma'$ is more general than $\sigma$" is written $\sigma' \sqsubseteq \sigma$: instantiate type variables in $\sigma'$ to give $\sigma$; quantify over type variables that are not free in $\sigma$, i.e., those that were bound in $\sigma$ and remain uninstantiated in the new type.

$$\forall\alpha\ .\ \alpha \longrightarrow \alpha \sqsubseteq \mathbf{int} \longrightarrow \mathbf{int} \qquad \alpha := \mathbf{int}$$

$$\forall\alpha\ .\ \alpha \longrightarrow \alpha \sqsubseteq \mathbf{bool} \longrightarrow \mathbf{bool} \qquad \alpha := \mathbf{bool}$$

$$\forall\alpha\ .\ \alpha \longrightarrow \alpha \not\sqsubseteq \mathbf{bool} \longrightarrow \mathbf{int} \qquad \text{Not consistent}$$

$$\forall\alpha\ .\ \alpha \longrightarrow \alpha \sqsubseteq \forall\alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow (\alpha \longrightarrow \alpha) \qquad \alpha := \forall\alpha\ .\ \alpha \longrightarrow \alpha$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var}$$

As in simply-typed lambda calculus:

The type of variable $x$ is what our enviroment says it is, a type scheme in general

Damas and Milner called this the tautology rule

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app}$$

As in simply-typed lambda calculus:

If $e_1 : \tau_2 \longrightarrow \tau_1$ is a function and $e_2 : \tau_2$ is its argument, applying $e_2$ to $e_1$ gives $\tau_1$

Note that these are just types, not type schemes

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

Similar to simply-typed lambda calculus:

If we take $x : \tau_1$ and conclude $e : \tau_2$, then $\lambda x\ .\ e : \tau_1 \longrightarrow \tau_2$

All just types, not type schemes

Big difference: the $\lambda$ term does not constrain the type of $x$. This is the inference part

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= $\tau$ | Type |
| | | **let** $x$ = $e$ **in** $e$ | Let binding | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let}$$

Rule for **let** like app + abs: $(\lambda x\ .\ e_2)\ e_1 : \tau_2$ if $e_1 : \tau_1$ and assuming $x : \tau_1$ gives $e_2 : \tau_2$

Important distinction: $e_1$ and the variable $x$ may have a *type scheme*

The key difference between $\lambda$ and **let** in the Hindley-Milner system: $\lambda$ arguments must be types; **let** variables may have *type schemes*.

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ **=** $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let}$$

```
Prelude> :t   let f = \x -> x in (f True , f 'a')
let f = \x -> x in (f True , f 'a') :: (Bool, Char)

Prelude> :t   (\f -> (f True , f 'a')) (\x -> x)
    Couldn't match expected type Bool with actual type Char
    In the first argument of f, namely 'a'
```

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= $\tau$ | Type |
| | | **let** $x$ = $e$ **in** $e$ | Let binding | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let}$$

```
# let f = fun x -> x in (f true, f 'a');;
- : bool * char = (true, 'a')

# (fun f -> (f true, f 'a')) (fun x -> x);;
Error: This expression has type char but expected an expr of type bool
```

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst}$$

The rule for instantiating type schemes:

An expression with a type scheme $\sigma'$ can be made more specific

This is how type schemes (e.g., for variables) can be reduced to types

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

Generalization rule:

You can always introduce a fresh type variable (not already appearing free in the environment) to the type of an expression.

This may create or just extend a type scheme

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ ...\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

FV(Γ): set of free type variables in an environment

$$\text{FV}(\alpha) = \{\alpha\}$$

$$\text{FV}(C\tau_1 \ldots \tau_k) = \text{FV}(\tau_1) \cup \cdots \cup \text{FV}(\tau_k)$$

$$\text{FV}(\forall \alpha\ .\ \sigma) = \text{FV}(\sigma) - \{\alpha\}$$

$$\text{FV}(\Gamma, x : \sigma) = \text{FV}(\sigma) \cup \text{FV}(\Gamma)$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \longrightarrow \alpha$?

$$\Gamma \vdash \text{id n} : ?$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\frac{\Gamma \vdash \text{id} : ? \qquad \to ? \qquad\qquad\qquad\qquad \Gamma \vdash \text{n} : ?}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\frac{\Gamma \vdash \text{id} : ? \quad \to ? \qquad \dfrac{\text{n} : ? \quad \in \Gamma}{\Gamma \vdash \text{n} : ?}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\frac{\Gamma \vdash \text{id} : ? \quad \to ? \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : ?}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\frac{\Gamma \vdash \text{id} : ? \quad \to ? \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id}\,\text{n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\frac{\Gamma \vdash \text{id} : \textbf{int} \to ? \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\Gamma \vdash \text{id} : ? \qquad ? \qquad \sqsubseteq \textbf{int} \to ?}{\Gamma \vdash \text{id} : \textbf{int} \to ?}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : ?}\ \text{var} \qquad ? \qquad \sqsubseteq \textbf{int} \to ?}{\Gamma \vdash \text{id} : \textbf{int} \to ?}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : \forall \alpha\ .\ \alpha \to \alpha}\ \text{var} \quad ? \quad \sqsubseteq \textbf{int} \to ?}{\Gamma \vdash \text{id} : \textbf{int} \to ?}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : \forall \alpha\ .\ \alpha \to \alpha}\ \text{var} \qquad \forall \alpha\ .\ \alpha \to \alpha \sqsubseteq \textbf{int} \to ?}{\Gamma \vdash \text{id} : \textbf{int} \to ?}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= $\tau$ | Type |
| | | **let** $x$ = $e$ **in** $e$ | Let binding | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : \forall \alpha\ .\ \alpha \to \alpha}\ \text{var} \qquad \forall \alpha\ .\ \alpha \to \alpha \sqsubseteq \textbf{int} \to \textbf{int}}{\Gamma \vdash \text{id} : \textbf{int} \to ?}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

$$
\begin{array}{rlr}
e & ::= & x \quad \text{Variable} \\
& & e\ e \quad \text{Application} \\
& & \lambda x\ .\ e \quad \text{Abstraction} \\
& & \textbf{let}\ x = e\ \textbf{in}\ e \quad \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau & ::= & C\ \tau\ \tau\ \ldots\ \tau \quad \text{Type application} \\
& & \alpha \quad \text{Type variable} \\
\sigma & ::= & \tau \quad \text{Type} \\
& & \forall\ \alpha\ .\ \sigma \quad \text{Quantified Type}
\end{array}
$$

$$
\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var}
\qquad
\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app}
\qquad
\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}
$$

$$
\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let}
\qquad
\frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst}
\qquad
\frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}
$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$
\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : \forall \alpha\ .\ \alpha \to \alpha}\ \text{var} \qquad \forall \alpha\ .\ \alpha \to \alpha \sqsubseteq \textbf{int} \to \textbf{int}}{\Gamma \vdash \text{id} : \textbf{int} \to \textbf{int}}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : ?}\ \text{app}
$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

What is the type of "id n" assuming $\Gamma$ = n : **int**, id : $\forall \alpha\ .\ \alpha \to \alpha$?

$$\dfrac{\dfrac{\dfrac{\text{id} : \forall \alpha\ .\ \alpha \to \alpha \in \Gamma}{\Gamma \vdash \text{id} : \forall \alpha\ .\ \alpha \to \alpha}\ \text{var} \qquad \forall \alpha\ .\ \alpha \to \alpha \sqsubseteq \textbf{int} \to \textbf{int}}{\Gamma \vdash \text{id} : \textbf{int} \to \textbf{int}}\ \text{inst} \qquad \dfrac{\text{n} : \textbf{int} \in \Gamma}{\Gamma \vdash \text{n} : \textbf{int}}\ \text{var}}{\Gamma \vdash \text{id n} : \textbf{int}}\ \text{app}$$

## Hindley-Milner Type System Deductive Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \longrightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \longrightarrow \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ ...\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\frac{\Gamma \vdash \lambda x.x : ? \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ ...\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\frac{\dfrac{\Gamma \vdash \lambda x.x : ? \qquad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : ?}\ \text{gen} \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\Gamma, x : \alpha \vdash x : ?}{\Gamma \vdash \lambda x.x : ?}\ \text{abs}}{\Gamma \vdash \lambda x.x : ?}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \Gamma, i : ? \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

Fundamental trick: pick a new type variable $\alpha$ as the type of the argument

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : ? \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs}}{\Gamma \vdash \lambda x.x : ?}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs}}{\Gamma \vdash \lambda x.x : ?}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs}}{\Gamma \vdash \lambda x.x : ?}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \Gamma, i : ? \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : ?}\ \text{gen} \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \Gamma, i : ? \qquad \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

Generalize the type of $i$ into a type scheme; we will need this later

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}\ \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \to ? \qquad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

Both environments get the same type scheme for $i$; they will instantiate it differently

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \qquad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app} \qquad \dfrac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \qquad ? \qquad \sqsubseteq ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{ins}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\dfrac{\dfrac{i : ? \qquad \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{var}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{ins} \qquad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \qquad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \qquad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$$\dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{var}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{ins} \qquad ? \quad \sqsubseteq\ ?$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\, .\, e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{ins} \qquad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\, i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\, i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \quad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ?}\ \text{ins} \quad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \quad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\, i : ?}\ \text{app}$$

$$\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\, i : ?\ \text{let}$$

A key step: we can instantiate the type scheme however we like

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : ? \qquad \to ?}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$$\dfrac{\dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \alpha \to \alpha}\ \text{ins}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}\ \text{ins}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \alpha \to \alpha}$$

$$\frac{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : (\alpha \to \alpha) \to ? \qquad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}$$

$$\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?\ \text{let}$$

We now know a little about the the type of $i$ we want on the left

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\dfrac{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : ?}{? \quad \sqsubseteq (\alpha\to\alpha)\to?}\ \text{inst} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}\ \text{ins}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : (\alpha\to\alpha)\to?}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}\ \ \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : (\alpha\to\alpha)\to ?}\ \text{inst}\ \ ? \sqsubseteq (\alpha\to\alpha)\to ? \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \alpha\to\alpha}\ \text{ins}\ \ \forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\,i : ?}\ \text{app}$$

$$\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\,i : ?\ \ \text{let}$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq (\alpha \to \alpha) \to (\alpha \to \alpha)}\ \text{inst} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}\ \text{ins}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : (\alpha \to \alpha) \to ?} \qquad \dfrac{}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \alpha \to \alpha}$$

$$\frac{}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\, i : ?}\ \text{app}$$

$$\frac{}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\, i : ?}\ \text{let}$$

This time, we instantiate the type scheme differently

## Hindley-Milner Type System Deductive Rules

$$
\begin{array}{rlr}
e & ::= & x & \text{Variable} \\
 & & e\ e & \text{Application} \\
 & & \lambda x\ .\ e & \text{Abstraction} \\
 & & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau & ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & & \alpha & \text{Type variable} \\
\sigma & ::= & \tau & \text{Type} \\
 & & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq (\alpha \to \alpha) \to (\alpha \to \alpha)}\ \text{inst} \qquad \dfrac{\dfrac{i : \forall \alpha.\alpha \to \alpha \in \Gamma, i : \forall \alpha.\alpha \to \alpha}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \forall \alpha.\alpha \to \alpha}\ \text{var}}{\forall \alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}\ \text{ins}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : (\alpha \to \alpha) \to (\alpha \to \alpha) \qquad \Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i : \alpha \to \alpha}\ \text{app}}{\Gamma, i : \forall \alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{let}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}$$

## Hindley-Milner Type System Deductive Rules

$$
\begin{array}{rlr}
e & ::= & x & \text{Variable} \\
 & & e\ e & \text{Application} \\
 & & \lambda x\ .\ e & \text{Abstraction} \\
 & & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau & ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & & \alpha & \text{Type variable} \\
\sigma & ::= & \tau & \text{Type} \\
 & & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$
\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var}
\qquad
\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app}
\qquad
\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}
$$

$$
\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let}
\qquad
\frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst}
\qquad
\frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}
$$

$$
\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}\ \ \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \text{gen}
\qquad
\dfrac{\dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\forall\alpha.\alpha\to\alpha \sqsubseteq (\alpha\to\alpha)\to(\alpha\to\alpha)}\ \text{inst}
\qquad
\dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}\ \text{ins}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \alpha\to\alpha}\ \text{app}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\,i : \alpha\to\alpha}\ \text{let}
$$

$$
\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\,i : ?
$$

## Hindley-Milner Type System Deductive Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\, x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \sigma \quad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau}\ \text{let} \qquad \frac{\Gamma \vdash e : \sigma' \quad \sigma' \sqsubseteq \sigma}{\Gamma \vdash e : \sigma}\ \text{inst} \qquad \frac{\Gamma \vdash e : \sigma \quad \alpha \notin \text{FV}(\Gamma)}{\Gamma \vdash e : \forall \alpha\ .\ \sigma}\ \text{gen}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, x : \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs}}{\Gamma \vdash \lambda x.x : \forall \alpha.\alpha \to \alpha}\ \alpha \notin \text{FV}(\Gamma)\ \text{gen} \qquad \dfrac{\dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\forall\alpha.\alpha\to\alpha \sqsubseteq (\alpha\to\alpha)\to(\alpha\to\alpha)}\ \text{inst} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma, i : \forall\alpha.\alpha\to\alpha}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \forall\alpha.\alpha\to\alpha}\ \text{var}}{\forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}\ \text{inst}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i : \alpha\to\alpha}\ \text{ins}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\, i : \alpha\to\alpha}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\, i : \alpha\to\alpha}\ \text{let}$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= $\tau$ | Type |
| | | **let** $x$ **=** $e$ **in** $e$ | Let binding | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

Clément, Despeyroux, Despeyroux, and Kahn. *A Simple Applicative Language: Mini-ML*, LISP and Functional Programming, 1986.

Expressions, types, and type schemes as before

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var}$$

Type instantiation now done only as part of the *var* rule

All expressions, including variables, now only take on types

Type schemes only appear in $\Gamma$ and instantiated when their variable appears

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \rightarrow \tau_2}\ \text{abs}$$

*app* and *abs* rules as before

All expressions take on types

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

A variable's type is generalized to a scheme when the *let* rule adds it to the context

The $\text{gen}(\Gamma, \tau)$ function generalizes the type $\tau$ to a scheme that quantifies all type variables that are free in $\tau$ and not free in $\Gamma$:

$$\text{gen}(\Gamma, \tau) = \begin{cases} \forall \alpha_1\ .\ \ldots\ .\ \forall \alpha_n\ .\ \tau & \text{when } \text{FV}(\tau) - \text{FV}(\Gamma) = \{\alpha_1, \ldots, \alpha_n\} \\ \tau & \text{when } \text{FV}(\tau) - \text{FV}(\Gamma) = \emptyset \end{cases}$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

One rule each for variables, applications, abstractions, and let bindings

Proof tree structure is now set by the syntax of the program

No choices about where and how many inst/gen rules to apply

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\frac{\Gamma \vdash \lambda x.x : ? \qquad \Gamma, i : \text{gen}(\Gamma, ?\quad) \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\frac{\dfrac{\Gamma, x : ? \vdash x : ?}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \Gamma, i : \text{gen}(\Gamma, ?\quad) \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : ? \in \Gamma \quad ? \sqsubseteq ?}{\Gamma, x : ? \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \Gamma, i : \text{gen}(\Gamma, ?\quad) \vdash i\ i : ?}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : ? \in \Gamma \quad ? \sqsubseteq ?}{\Gamma, x : ? \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\Gamma_1 \vdash i : ? \qquad \to ? \qquad \Gamma_1 \vdash i : ?}{\Gamma, i : \text{gen}(\Gamma, ?\quad) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?\quad)$

Just an abbreviation

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{x : ? \in \Gamma \quad ? \sqsubseteq ?}{\Gamma, x : ? \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ? \to ?}\ \text{var} \qquad \Gamma_1 \vdash i : ?}{\Gamma, i : \text{gen}(\Gamma, ?) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?)$

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : ? \in \Gamma \quad ? \sqsubseteq ?}{\Gamma, x : ? \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ? \to ?}\ \text{var} \qquad \dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, ?) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?)$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : ? \in \Gamma \quad ? \sqsubseteq ?}{\Gamma, x : ? \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ? \to ?}\ \text{var} \qquad \dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, ?) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?)$

Proof structure set by syntax. Start with unconstrained argument type.

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq\ ?}{\Gamma, x : \alpha \vdash x :\ ?}\ \text{var}}{\Gamma \vdash \lambda x.x :\ ?}\ \text{abs} \qquad \dfrac{\dfrac{i :\ ? \quad \in \Gamma_1 \quad ?\ \sqsubseteq\ ?}{\Gamma_1 \vdash i :\ ? \quad \to ?}\ \text{var} \qquad \dfrac{i :\ ? \quad \in \Gamma_1 \quad ?\ \sqsubseteq\ ?}{\Gamma_1 \vdash i :\ ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, ?\ ) \vdash i\ i :\ ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i :\ ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?\ )$

Choose a fresh type variable; this is the most general type of an argument

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : ?}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \quad \in \Gamma_1 \quad ? \quad \sqsubseteq ?}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : ? \quad \in \Gamma_1 \quad ? \quad \sqsubseteq ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, ?\ ) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?\ \ )$

Instantiate without change for now

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : ?}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \qquad \in \Gamma_1}{? \qquad \sqsubseteq ?}\ \text{var} \qquad \dfrac{i : ? \qquad \in \Gamma_1}{? \qquad \sqsubseteq ?}\ \text{var}}{\Gamma_1 \vdash i : ? \qquad \to ? \qquad\qquad \Gamma_1 \vdash i : ?}\ \text{app}}{\Gamma, i : \text{gen}(\Gamma, ? \qquad ) \vdash i\ i : ?}\ \text{let}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ? \qquad )$

Type of body

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ? \to ?}\ \text{var} \qquad \dfrac{i : ? \in \Gamma_1 \quad ? \sqsubseteq ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, ?) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, ?)$

Type of $i$, the $\lambda$ expression

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \quad \in \Gamma_1 \qquad ? \quad \sqsubseteq ?}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : ? \quad \in \Gamma_1 \qquad ? \quad \sqsubseteq ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \text{gen}(\Gamma, \alpha \to \alpha) \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \text{gen}(\Gamma, \alpha \to \alpha)$

Generalize the type of $i$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : ? \quad \in \Gamma_1 \quad ? \quad \sqsubseteq \ ?}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : ? \quad \in \Gamma_1 \quad ? \quad \sqsubseteq \ ?}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha \to \alpha \quad \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Type scheme for $i$: generalize by quantifying the free type variables in $\alpha \to \alpha$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Type scheme for $i$ is in the context $\Gamma_1$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ ...\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{\dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq\ ?}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq\ \textbf{?}}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha \to \alpha \quad \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Instantiate the type of the argument $e_2$: $i$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq ?}{\Gamma_1 \vdash i : ? \quad \to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}{\Gamma_1 \vdash i : ?}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha \to \alpha \quad \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Instantiate to the type $\alpha \to \alpha$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq ?}{\Gamma_1 \vdash i : (\alpha \to \alpha) \to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha \to \alpha \in \Gamma_1 \quad \forall\alpha.\alpha \to \alpha \sqsubseteq \alpha \to \alpha}{\Gamma_1 \vdash i : \alpha \to \alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha \to \alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Sets the type of the argument to $i\ i$

## Hindley-Milner Type System Syntax-Directed Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x\ .\ e & \text{Abstraction} \\
 & \textbf{let}\ x = e\ \textbf{in}\ e & \text{Let binding}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & C\ \tau\ \tau\ \ldots\ \tau & \text{Type application} \\
 & \alpha & \text{Type variable} \\
\sigma ::= & \tau & \text{Type} \\
 & \forall\ \alpha\ .\ \sigma & \text{Quantified Type}
\end{array}
$$

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq \textbf{?}}{\Gamma_1 \vdash i : (\alpha\to\alpha)\to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}{\Gamma_1 \vdash i : \alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

A Unification Problem: $\forall\alpha\ .\ \alpha \to \alpha \sqsubseteq (\alpha \to \alpha) \to ?$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ ::= | $x$ | Variable | $\tau$ ::= | $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= | $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq (\alpha\to\alpha)\to(\alpha\to\alpha)}{\Gamma_1 \vdash i : (\alpha\to\alpha)\to ?}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}{\Gamma_1 \vdash i : \alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \quad \vdash i\ i : ?}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Solution: $\alpha := \alpha \to \alpha$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= $x$ | Variable | $\tau$ | ::= $C\ \tau\ \tau\ \ldots\ \tau$ | Type application |
| | $e\ e$ | Application | | $\alpha$ | Type variable |
| | $\lambda x\ .\ e$ | Abstraction | $\sigma$ | ::= $\tau$ | Type |
| | **let** $x$ = $e$ **in** $e$ | Let binding | | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall\{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq (\alpha\to\alpha)\to(\alpha\to\alpha)}{\Gamma_1 \vdash i : (\alpha\to\alpha)\to(\alpha\to\alpha)}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}{\Gamma_1 \vdash i : \alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \quad \vdash i\ i : \alpha\to\alpha}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : ?}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

Now we know the type of the body $i\ i$

## Hindley-Milner Type System Syntax-Directed Rules

| | | | | | |
|---|---|---|---|---|---|
| $e$ | ::= | $x$ | Variable | $\tau$ ::= $C\ \tau\ \tau\ ...\ \tau$ | Type application |
| | | $e\ e$ | Application | $\alpha$ | Type variable |
| | | $\lambda x\ .\ e$ | Abstraction | $\sigma$ ::= $\tau$ | Type |
| | | **let** $x$ = $e$ **in** $e$ | Let binding | $\forall\ \alpha\ .\ \sigma$ | Quantified Type |

$$\frac{x : \sigma \in \Gamma \quad \sigma \sqsubseteq \tau}{\Gamma \vdash x : \tau}\ \text{var} \qquad \frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ \text{app} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda\ x\ .\ e : \tau_1 \to \tau_2}\ \text{abs}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \text{gen}(\Gamma, \tau_1) \vdash e_2 : \tau_2}{\Gamma \vdash \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \tau_2}\ \text{let} \qquad \text{gen}(\Gamma, \tau) = \forall \{\text{FV}(\tau) - \text{FV}(\Gamma)\}\ .\ \tau$$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma \quad \alpha \sqsubseteq \alpha}{\Gamma, x : \alpha \vdash x : \alpha}\ \text{var}}{\Gamma \vdash \lambda x.x : \alpha \to \alpha}\ \text{abs} \qquad \dfrac{\dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq (\alpha\to\alpha)\to(\alpha\to\alpha)}{\Gamma_1 \vdash i : (\alpha\to\alpha)\to(\alpha\to\alpha)}\ \text{var} \qquad \dfrac{i : \forall\alpha.\alpha\to\alpha \in \Gamma_1 \quad \forall\alpha.\alpha\to\alpha \sqsubseteq \alpha\to\alpha}{\Gamma_1 \vdash i : \alpha\to\alpha}\ \text{var}}{\Gamma, i : \forall\alpha.\alpha\to\alpha \vdash i\ i : \alpha\to\alpha}\ \text{app}}{\Gamma \vdash \textbf{let}\ i = \lambda x.x\ \textbf{in}\ i\ i : \alpha\to\alpha}\ \text{let}$$

$\Gamma_1 = \Gamma, i : \forall\alpha\ .\ \alpha \to \alpha$

The expression is well-typed