Type Checking Cubex

Ross Tate

CS 4120

Contexts in Cubex

- Ψ – Class/Interface context
  - Specifies methods, inheritance, and classiness
- Θ – Kind context
  - Indicates what type variables are in scope
- Δ – Function context
  - Indicates the type schemes of functions in scope
- Γ – Type context
  - Indicates the types of variables in scope

Subtyping

$\Psi \mid \Theta \vdash \tau <: \tau \qquad \Psi \vdash \nu\langle\Theta\rangle \textbf{ extends } \tau$

$$\frac{}{\Psi \mid \Theta \vdash \nu <: \nu} \qquad \frac{}{\Psi \mid \Theta \vdash \bot <: \tau} \qquad \frac{}{\Psi \mid \Theta \vdash \tau <: \top}$$

$$\frac{\Psi \mid \Theta \vdash \tau_i <: \tau}{\Psi \mid \Theta \vdash \tau_1 \cap \tau_2 <: \tau} \qquad \frac{\Psi \mid \Theta \vdash \tau <: \tau_1 \quad \Psi \mid \Theta \vdash \tau <: \tau_2}{\Psi \mid \Theta \vdash \tau <: \tau_1 \cap \tau_2}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \vdash \tau_i <: \tau'_i \quad \text{and} \quad \Psi \mid \Theta \vdash \tau'_i <: \tau_i}{\Psi \mid \Theta \vdash \nu\langle\tau_1,\ldots,\tau_n\rangle <: \nu\langle\tau'_1,\ldots,\tau'_n\rangle} \qquad \frac{\Psi \mid \Theta \vdash \tau <: \tau'}{\Psi \mid \Theta \vdash \texttt{Iterable}\langle\tau\rangle <: \texttt{Iterable}\langle\tau'\rangle}$$

$$\frac{\begin{array}{c}\Psi \vdash \nu\langle\nu_1,\ldots,\nu_n\rangle \textbf{ extends } \tau' \\ \Psi \mid \Theta \vdash \tau'[\nu_1 \mapsto \tau_1,\ldots,\nu_n \mapsto \tau_n] <: \tau\end{array}}{\Psi \mid \Theta \vdash \nu\langle\tau_1,\ldots\tau_n\rangle <: \tau} \qquad \frac{\textbf{interface } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\} \text{ in } \Psi}{\Psi \vdash \nu\langle\Theta\rangle \textbf{ extends } \tau} \qquad \frac{\textbf{class } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\} \text{ in } \Psi}{\Psi \vdash \nu\langle\Theta\rangle \textbf{ extends } \tau}$$

Method Lookup

$\Psi \mid \Theta \vdash \tau.\nu : \sigma \qquad \Psi \vdash \nu\langle\Theta\rangle.\nu : \sigma$

$$\frac{\Psi \mid \Theta \vdash \tau'.\nu : \sigma \quad \Psi \mid \Theta \vdash \tau <: \tau'}{\Psi \mid \Theta \vdash \tau.\nu : \sigma} \qquad \frac{}{\Psi \mid \Theta \vdash \bot.\nu : \sigma}$$

$$\frac{\Psi \vdash \hat{\nu}\langle\nu_1,\ldots,\nu_n\rangle.\nu : \sigma}{\Psi \mid \Theta \vdash \hat{\nu}\langle\tau_1,\ldots,\tau_n\rangle.\nu : \sigma[\nu_1 \mapsto \tau_1,\ldots,\nu_n \mapsto \tau_n]}$$

$$\frac{\textbf{interface } \hat{\nu}\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\nu\sigma;\ldots\} \text{ in } \Psi}{\Psi \vdash \hat{\nu}\langle\Theta\rangle.\nu : \sigma} \qquad \frac{\textbf{class } \hat{\nu}\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\nu\sigma;\ldots\} \text{ in } \Psi}{\Psi \vdash \hat{\nu}\langle\Theta\rangle.\nu : \sigma}$$

Types

$\Psi \mid \Theta \vdash \tau \qquad \Psi \mid \Theta \vdash_\tau \tau \qquad \Psi \mid \Theta \vdash \Gamma \qquad \Psi \mid \Theta \vdash \sigma$

$$\frac{\Psi \mid \Theta \vdash_{\hat{\tau}} \tau}{\Psi \mid \Theta \vdash \tau} \qquad \frac{}{\Psi \mid \Theta \vdash_\top \top} \qquad \frac{}{\Psi \mid \Theta \vdash \bot} \qquad \frac{\nu \text{ in } \Theta}{\Psi \mid \Theta \vdash \nu}$$

$\nu_1 \ldots \nu_n$

$$\frac{\textbf{interface } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\} \text{ in } \Psi \quad \text{for all } i, \quad \Psi \mid \Theta \vdash \tau_i}{\Psi \mid \Theta \vdash_\top \nu\langle\tau_1,\ldots,\tau_n\rangle}$$

$\nu_1 \ldots \nu_n$

$$\frac{\textbf{class } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\ldots\} \text{ in } \Psi \quad \text{for all } i, \quad \Psi \mid \Theta \vdash \tau_i}{\Psi \mid \Theta \vdash_{\nu\langle\tau_1,\ldots,\tau_n\rangle} \nu\langle\tau_1,\ldots,\tau_n\rangle}$$

$$\frac{\begin{array}{c}\Psi \mid \Theta \vdash_{\hat{\tau}} \tau \qquad\qquad\qquad\qquad \Psi \mid \Theta \vdash_\top \tau' \\ \text{for all } \nu, \ \left(\begin{array}{c}\Psi \mid \Theta \vdash \tau <: \nu\langle\tau_1,\ldots,\tau_n\rangle \\ \text{and} \\ \Psi \mid \Theta \vdash \tau' <: \nu\langle\tau'_1,\ldots,\tau'_n\rangle\end{array}\right) \text{ implies } \left(\begin{array}{c}\Psi \mid \Theta \vdash \tau <: \nu\langle\tau'_1,\ldots,\tau'_n\rangle \\ \text{and} \\ \Psi \mid \Theta \vdash \tau' <: \nu\langle\tau_1,\ldots,\tau_n\rangle\end{array}\right) \\ \text{for all } \nu, \quad \Psi \mid \Theta \vdash \tau.\nu : \sigma \quad \text{and} \quad \Psi \mid \Theta \vdash \tau'.\nu : \sigma' \quad \text{implies} \quad \sigma = \sigma'\end{array}}{\Psi \mid \Theta \vdash_{\hat{\tau}} \tau \cap \tau'}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \vdash \tau_i}{\Psi \mid \Theta \vdash \nu_1 : \tau_1,\ldots,\nu_n : \tau_n} \qquad \frac{\Psi \mid \Theta, \hat{\Theta} \vdash \Gamma \quad \Psi \mid \Theta, \hat{\Theta} \vdash \tau}{\Psi \mid \Theta \vdash \langle\hat{\Theta}\rangle(\Gamma) : \tau}$$

Expressions

foo<Int>([1,2,3]) : Integer

Δ contains foo<P>(Iterable<P>) : P

$\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e : \tau$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e : \tau \quad \Psi \mid \Theta \vdash \tau <: \tau'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e : \tau'}$$

$$\frac{\nu : \tau \text{ in } \Gamma}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash \nu : \tau}$$

$$\frac{\begin{array}{c}\nu\langle\nu_1,\ldots,\nu_m\rangle(\hat{\nu}_1 : \hat{\tau}_1,\ldots,\hat{\nu}_n : \hat{\tau}_n) : \tau \text{ in } \Delta \\ \text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e_i : \tau'_i[\nu_1 \mapsto \tau_1,\ldots,\nu_m \mapsto \tau_m]\end{array}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash \nu\langle\tau_1,\ldots,\tau_m\rangle(e_1,\ldots,e_n) : \tau[\nu_1 \mapsto \tau_1,\ldots,\nu_m \mapsto \tau_m]}$$

$$\frac{\begin{array}{c}\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e : \hat{\tau} \\ \Psi \mid \Theta \vdash \hat{\tau}.\nu : \langle\nu_1,\ldots,\nu_m\rangle(\hat{\nu}_1 : \hat{\tau}_1,\ldots,\hat{\nu}_n : \hat{\tau}_n) : \tau \\ \text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e_i : \hat{\tau}_i[\nu_1 \mapsto \tau_1,\ldots,\nu_m \mapsto \tau_m]\end{array}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e.\nu\langle\tau_1,\ldots,\tau_m\rangle(e_1,\ldots,e_n) : \tau[\nu_1 \mapsto \tau_1,\ldots,\nu_m \mapsto \tau_m]}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e_i : \tau}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash [e_1,\ldots,e_n] : \texttt{Iterable}\langle\tau\rangle} \qquad \frac{\text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e_i : \texttt{Iterable}\langle\tau\rangle}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash e_1 +\!\!+\, e_2 : \texttt{Iterable}\langle\tau\rangle}$$

$$\frac{}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash \textbf{true} : \texttt{Boolean}\langle\rangle} \qquad \frac{}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash \textbf{false} : \texttt{Boolean}\langle\rangle}$$

$$\frac{}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash n : \texttt{Integer}\langle\rangle} \qquad \frac{}{\Psi \mid \Theta \mid \Delta \mid \Gamma \vdash \texttt{"string"} : \texttt{String}\langle\rangle}$$

Statements

$\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s : \hat{\Gamma}'$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s : \hat{\Gamma}', \nu : \tau, \hat{\Gamma}''}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s : \hat{\Gamma}', \hat{\Gamma}''}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}_{i-1} \vdash s_i : \hat{\Gamma}_i}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}_0 \vdash \{s_1 \ldots s_n\} : \hat{\Gamma}_n}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \tau}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash \nu := e; : \hat{\Gamma}, \nu : \tau} \qquad \frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma}, \nu : \tau, \hat{\Gamma}' \vdash e : \tau'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}, \nu : \tau, \hat{\Gamma}' \vdash \nu := e; : \hat{\Gamma}, \nu : \tau', \hat{\Gamma}'}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Boolean}\langle\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s_1 : \hat{\Gamma}' \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s_2 : \hat{\Gamma}'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash \textbf{if } (e)\ s_1 \textbf{ else } s_2 : \hat{\Gamma}'}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Boolean}\langle\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash s : \hat{\Gamma}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash \textbf{while } (e)\ s : \hat{\Gamma}}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Iterable}\langle\tau\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}, \nu : \tau \vdash s : \hat{\Gamma}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash \textbf{for } (\nu \textbf{ in } e)\ s : \hat{\Gamma}}$$

Returns

$\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \Gamma \vdash^b_\tau s : \Gamma$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{true}}_\tau s : \hat{\Gamma}'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{false}}_\tau s : \hat{\Gamma}'} \qquad \frac{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau s : \hat{\Gamma}', \nu : \hat{\tau}, \hat{\Gamma}''}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau s : \hat{\Gamma}', \hat{\Gamma}''}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}_{i-1} \vdash^{b_i}_\tau s_i : \hat{\Gamma}_i}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}_0 \vdash^{b_n}_\tau \{s_1 \ldots s_n\} : \hat{\Gamma}_n}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash_\tau e : \hat{\tau}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{false}}_\tau \nu := e; : \hat{\Gamma}, \nu : \hat{\tau}} \qquad \frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma}, \nu : \hat{\tau}, \hat{\Gamma}' \vdash_\tau e : \hat{\tau}'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}, \nu : \hat{\tau}, \hat{\Gamma}' \vdash^{\textbf{false}}_\tau \nu := e; : \hat{\Gamma}, \nu : \hat{\tau}', \hat{\Gamma}'}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Boolean}\langle\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau s_1 : \hat{\Gamma}' \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau s_2 : \hat{\Gamma}'}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau \textbf{if } (e)\ s_1 \textbf{ else } s_2 : \hat{\Gamma}'}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Boolean}\langle\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^b_\tau s : \hat{\Gamma}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{false}}_\tau \textbf{while } (e)\ s : \hat{\Gamma}}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \texttt{Iterable}\langle\hat{\tau}\rangle \quad \Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma}, \nu : \hat{\tau} \vdash^b_\tau s : \hat{\Gamma}}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{false}}_\tau \textbf{for } (\nu \textbf{ in } e)\ s : \hat{\Gamma}}$$

$$\frac{\Psi \mid \Theta \mid \Delta \mid \Gamma, \hat{\Gamma} \vdash e : \tau}{\Psi \mid \Theta \mid \Delta \mid \Gamma \mid \hat{\Gamma} \vdash^{\textbf{true}}_\tau \textbf{return } e; : \hat{\Gamma}'}$$

Class/Interface

$\Psi \mid \Delta \mid \Gamma \vdash i : \Psi' \qquad \Psi \mid \Delta \mid \Gamma \vdash c : \Psi' \mid \Delta'$

$$\frac{\begin{array}{c} \Psi \mid \Theta \vdash_\top \tau \\ \Psi' = \textbf{interface } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\nu_1\sigma_1; \ldots; \nu_n\sigma_n;\} \\ \text{for all } i, \quad \Psi, \Psi' \mid \Theta \vdash \sigma_i \\ \text{for all } i, \quad \Psi \mid \Theta \vdash \tau.\nu_i : \sigma \quad \text{implies} \quad \sigma = \sigma_i \end{array}}{\Psi \mid \Delta \mid \Gamma \vdash \textbf{interface } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\textbf{fun } \nu_1\sigma_1; \ldots; \textbf{fun } \nu_n\sigma_n;\} : \Psi'}$$

$$\frac{\begin{array}{c} \Psi \mid \Theta \vdash_\tau \tau \\ \Psi' = \textbf{class } \nu\langle\Theta\rangle \textbf{ extends } \tau\ \{\nu_1\langle\Theta_1\rangle(\Gamma_1); \ldots; \nu_n\langle\Theta_n\rangle(\Gamma_n);\} \qquad \Delta' = \nu\langle\Theta\rangle(\hat{\Gamma}) : \nu\langle\Theta\rangle \\ \Psi, \Psi' \mid \Theta \vdash \hat{\Gamma} \qquad \hat{\Gamma}_0 = \hat{\Gamma} \qquad \text{for all } i, \quad \Psi, \Psi' \mid \Theta \mid \Delta, \Delta' \mid \Gamma \mid \hat{\Gamma}_{i-1} \vdash s_i : \hat{\Gamma}_i \\ \hat{\tau}(e_1, \ldots, e_k) = \top() \quad \text{or} \quad \Psi, \Psi' \mid \Theta \mid \Delta, \Delta' \mid \Gamma, \hat{\Gamma}_n \vdash \hat{\tau}(e_1, \ldots, e_k) : \hat{\tau} \\ \Delta'' = \Delta, \Delta', \nu_1\langle\Theta_1\rangle(\Gamma_1), \ldots, \nu_n\langle\Theta_n\rangle(\Gamma_n) \\ \text{for all } i, \quad \Psi, \Psi' \mid \Theta \vdash \sigma_i \quad \sigma_i = \langle\Theta_i\rangle(\Gamma_i) : \tau_i \quad \Psi, \Psi' \mid \Theta, \Theta_i \mid \Delta'' \mid \Gamma, \hat{\Gamma}_n \mid \Gamma_i \vdash^{\textbf{true}}_{\tau_i} \hat{s}_i : \Gamma_i' \\ \text{for all } \hat{\nu}\ \sigma, \quad \Psi \mid \Theta \vdash \tau.\hat{\nu} : \sigma \quad \text{implies} \quad \text{exists } i, \quad \nu_i = \hat{\nu} \quad \sigma_i = \sigma \end{array}}{\Psi \mid \Delta \mid \Gamma \vdash \textbf{class } \nu\langle\Theta\rangle(\hat{\Gamma}) \textbf{ extends } \tau\ \{s_1 \ldots s_m \textbf{ super}(e_1, \ldots, e_k);\ \textbf{fun } \nu_1\sigma_1\ \hat{s}_1\ \ldots\ \textbf{fun } \nu_n\sigma_n\ \hat{s}_n\} : \Psi' \mid \Delta'}$$

Programs

$\Psi \mid \Delta \mid \Gamma \vdash p \qquad \vdash p$

$$\frac{\Psi \mid \varnothing \mid \Delta \mid \Gamma \mid \varnothing \vdash^{\textbf{true}}_{\texttt{String}} s : \hat{\Gamma}}{\Psi \mid \Delta \mid \Gamma \vdash s}$$

$$\frac{\hat{\Gamma}_0 = \varnothing \quad \text{for all } i, \quad \Psi \mid \varnothing \mid \Delta \mid \Gamma \mid \hat{\Gamma}_{i-1} \vdash^{b_i}_{\texttt{String}} s_i : \hat{\Gamma}_i \quad \Psi \mid \Delta \mid \Gamma, \hat{\Gamma}_n \vdash p}{\Psi \mid \Delta \mid \Gamma \vdash s_1 \ldots s_n\ p}$$

$$\frac{\begin{array}{c} \Delta' = \Delta, \nu_1\langle\Theta_1\rangle(\Gamma_1) : \tau_1, \ldots, \nu_n\langle\Theta_n\rangle(\Gamma_n) : \tau_n \\ \text{for all } i, \quad \Psi \mid \Theta_i \vdash \Gamma_i \quad \Psi \mid \Theta_i \vdash \tau_i \quad \Psi \mid \Theta_i \mid \Delta' \mid \Gamma \mid \Gamma_i \vdash^{\textbf{true}}_{\tau_i} s_i : \hat{\Gamma}_i \\ \Psi \mid \Delta' \mid \Gamma \vdash p \end{array}}{\Psi \mid \Delta \mid \Gamma \vdash \textbf{fun } \nu_1\langle\Theta_1\rangle(\Gamma_1) : \tau_1\ s_1 \ldots \textbf{fun } \nu_n\langle\Theta_n\rangle(\Gamma_n) : \tau_n\ s_n\ p}$$

$$\frac{\Psi \mid \Delta \mid \Gamma \vdash i : \Psi' \quad \Psi, \Psi' \mid \Delta \mid \Gamma \vdash p}{\Psi \mid \Delta \mid \Gamma \vdash i\ p} \qquad \frac{\Psi \mid \Delta \mid \Gamma \vdash c : \Psi' \mid \Delta' \quad \Psi, \Psi' \mid \Delta, \Delta' \mid \Gamma \vdash p}{\Psi \mid \Delta \mid \Gamma \vdash c\ p}$$

$$\frac{\Psi_0 \mid \Delta_0 \mid \varnothing \vdash p}{\vdash p}$$

Principal Types

- Expression *e* has principal type τ in some context if
  - (context) ⊢ *e* : τ holds
  - for all τ', if (context) ⊢ *e* : τ' holds then (context) ⊢ τ <: τ'
- In other words, while *e* may have many types, τ is the most precise one.
- Java, C#, and Scala do *not* have principal types
  - OCaml and Haskell have principal type *schemes*

Cubex has principal types!!!

The Empty Iterable

- What is the principal type of [] in Cubex?

Covariance!
$$\frac{\bot <: \tau}{\texttt{Iterable}\langle\bot\rangle <: \texttt{Iterable}\langle\tau\rangle}$$

Cubex has a bottom type!!!

Appending Iterables

- What is the principal type of e_1 ++ e_2 in Cubex?
  - Given each e_i has principal type τ_i

τ_i <: Iterable<τ_i'>

e_1 ++ e_2 : Iterable<$\tau_1' \sqcup \tau_2'$>

Cubex has principal instantiations!!!

Join – Most precise common supertype

- $\tau_1 \sqcup \tau_2$ denotes the *join* of τ_1 and τ_2 (if it exists)
  - $\tau_1 <: \tau_1 \sqcup \tau_2$ and $\tau_2 <: \tau_1 \sqcup \tau_2$
  - For any τ, $\tau_1 <: \tau$ and $\tau_2 <: \tau$ implies $\tau_1 \sqcup \tau_2 <: \tau$

class Foo extends A&B&D
class Bar extends D&C&A

Foo ⊔ Bar = A&D

Cubex has joins!!!

Covariant Arrays (not in Cubex)

Contravariance (not in Cubex)