System F

Stephen A. Edwards

Columbia University

Spring 2023

System F

Jean-Yvves Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Thèse d'état, Université Paris VII, 1972.

Polymorphic Lambda-Calculus

John Reynolds, Towards a Theory of Type Structure, Programming Symposium, LNCS 19, 1974.

These notes primarily from Pierce, *Types and Programming Languages*, Ch. 23, 2002.

System F

Jean-Yvves Girard, Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur. Thèse d'état, Université Paris VII, 1972.

Polymorphic Lambda-Calculus

John Reynolds, Towards a Theory of Type Structure, Programming Symposium, LNCS 19, 1974.

Parametric Polymorphism: Code truly accepts *any* type and does the same thing in each case. Examples: the identity function, the take-two, return-first function

System F is parametric

Ad-hoc Polymorphism: Code accepts arguments of different types, but may do something different. Examples: +, overloaded operators, Haskell typeclasses

System F Typing Rules

e	$::=$	x	Variable
		$e\ e$	Application
		$\lambda x : \tau\ .\ e$	Abstraction

As in simply-typed lambda calculus

Variable types are annotated with a type τ

All are terms; variables and λ and Λ abstractions are also values

System F Typing Rules

$$
\begin{array}{rll}
e & ::= & x & \text{Variable} \\
 & & e\ e & \text{Application} \\
 & & \lambda\, x : \tau\ .\ e & \text{Abstraction}
\end{array}
$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ .\ e) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

As in simply-typed lambda calculus

When type checking function body, add variable type to context

If passing an argument of type τ_1 makes the body a τ_2, it is a function of type $\tau_1 \longrightarrow \tau_2$

System F Typing Rules

e ::=	x	Variable
	$e\ e$	Application
	$\lambda x : \tau\ .\ e$	Abstraction

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ .\ e) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

As in simply-typed lambda calculus

Consult the context for the type of a variable

System F Typing Rules

$$
\begin{array}{rll}
e & ::= & x \qquad\qquad \text{Variable} \\
 & & e\ e \qquad\qquad \text{Application} \\
 & & \lambda\, x : \tau\ .\ e \qquad\qquad \text{Abstraction}
\end{array}
$$

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

As in simply-typed lambda calculus

Applying τ_1 to a function of type $\tau_1 \longrightarrow \tau_2$ produces a result of type τ_2

System F Typing Rules

$$
\begin{array}{rll}
e & ::= & x \quad \text{Variable} \\
 & & e\ e \quad \text{Application} \\
 & & \lambda x : \tau\ .\ e \quad \text{Abstraction}
\end{array}
\qquad
\begin{array}{rll}
\tau & ::= & \alpha \quad \text{Type Variable} \\
 & & \tau \longrightarrow \tau \quad \text{Function} \\
 & & \forall \alpha\ .\ \tau \quad \text{Quantified Type}
\end{array}
$$

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

Type variables | Function types | Universally quantified types

Unlike Hindley-Milner, no distinction between types and type schemes; no *let*-style polymorphism. Type constructors optional

Type variables are syntactically distinct from variables

System F Typing Rules

$$
\begin{array}{rcll}
e & ::= & x & \text{Variable} \\
 & & e\ e & \text{Application} \\
 & & \lambda x : \tau\ .\ e & \text{Abstraction} \\
 & & \Lambda \alpha\ .\ e & \text{Type Abstraction} \\
 & & e\ [\tau] & \text{Type Application}
\end{array}
\qquad
\begin{array}{rcll}
\tau & ::= & \alpha & \text{Type Variable} \\
 & & \tau \longrightarrow \tau & \text{Function} \\
 & & \forall \alpha\ .\ \tau & \text{Quantified Type}
\end{array}
$$

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

The big change: types are first-class objects

Functions can have type arguments $\Lambda \alpha\ .\ e$ Type arguments can be applied $e\ [\tau]$

Type arguments don't have types, e.g., not $\Lambda\, \alpha : \sigma\ .\ e$

Using Λ and `[]` is overkill: type variables distinct from variables makes types distinct from expressions

System F Typing Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x : \tau\ .\ e & \text{Abstraction} \\
 & \Lambda \alpha\ .\ e & \text{Type Abstraction} \\
 & e\ [\tau] & \text{Type Application}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & \alpha & \text{Type Variable} \\
 & \tau \longrightarrow \tau & \text{Function} \\
 & \forall \alpha\ .\ \tau & \text{Quantified Type}
\end{array}
$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs}$$

Taking a type as an argument $\Lambda \alpha\ .\ e$ means you must accept *any* type

The $\forall \alpha\ .\ \tau$ type means "I can take any type α." Like a function in type-land

The type variable is added to the context only to prevent type variable name capture

System F Typing Rules

$$
\begin{array}{rlr}
e ::= & x & \text{Variable} \\
 & e\ e & \text{Application} \\
 & \lambda x : \tau\ .\ e & \text{Abstraction} \\
 & \Lambda \alpha\ .\ e & \text{Type Abstraction} \\
 & e\ [\tau] & \text{Type Application}
\end{array}
\qquad
\begin{array}{rlr}
\tau ::= & \alpha & \text{Type Variable} \\
 & \tau \longrightarrow \tau & \text{Function} \\
 & \forall \alpha\ .\ \tau & \text{Quantified Type}
\end{array}
$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Typing context Γ only conveys types of variables, not values. So no "t-var for type variables" that gives the specific type passed to a $\forall$ type

Instead t-tapp substitutes a type variable α with its type argument τ_2

Like β-reduction at type checking time

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x$

$$\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x$

$$\frac{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : ? \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x$

$$\frac{\dfrac{\Gamma, \alpha, x : \alpha \vdash x : ?}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : ?}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x$

$$\frac{\dfrac{\dfrac{x : ? \in \Gamma, \alpha, x : \alpha}{\Gamma, \alpha, x : \alpha \vdash x : ?}\ \text{t-var}}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : ?}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, \alpha, x : \alpha}{\Gamma, \alpha, x : \alpha \vdash x : ?}\ \text{t-var}}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : ?}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x$

$$\frac{\dfrac{\dfrac{x : \alpha \in \Gamma, \alpha, x : \alpha}{\Gamma, \alpha, x : \alpha \vdash x : \alpha}\ \text{t-var}}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : ?}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x$

$$\dfrac{\dfrac{\dfrac{x : \alpha \in \Gamma, \alpha, x : \alpha}{\Gamma, \alpha, x : \alpha \vdash x : \alpha}\ \text{t-var}}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : \alpha \to \alpha}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x$

$$\frac{\dfrac{\dfrac{x : \alpha \in \Gamma, \alpha, x : \alpha}{\Gamma, \alpha, x : \alpha \vdash x : \alpha}\ \text{t-var}}{\Gamma, \alpha \vdash \lambda x : \alpha\ .\ x : \alpha \to \alpha}\ \text{t-abs} \qquad \alpha \notin \Gamma}{\Gamma \vdash (\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x) : \forall \alpha\ .\ \alpha \to \alpha}\ \text{t-tabs}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ \texttt{[}\tau\texttt{]} \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ \texttt{[}\tau_2\texttt{]} : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x) : \forall \alpha\ .\ \alpha \longrightarrow \alpha$

$$\Gamma \vdash (\Lambda \alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?$$

Let's assume there is a **`Bool`** type with a literal **`true`**

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ \texttt{[}\tau\texttt{]} \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ \texttt{[}\tau_2\texttt{]} : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \longrightarrow \alpha$

$$\frac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : ? \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha$

$$\frac{\dfrac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : ?}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : ?}\ \text{t-tapp} \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha$

$$\frac{\dfrac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : ?}\ \text{t-tapp} \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha$

$$\frac{\dfrac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : (\alpha \to \alpha)[\alpha := \texttt{Bool}]}\ \text{t-tapp} \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha$

$$\frac{\dfrac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \to \alpha}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : \texttt{Bool} \to \texttt{Bool}}\ \text{t-tapp} \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : ?}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ \texttt{[}\tau\texttt{]} \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ \texttt{[}\tau_2\texttt{]} : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic identity function: $\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \longrightarrow \alpha$

$$\frac{\dfrac{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x) : \forall\alpha\ .\ \alpha \longrightarrow \alpha}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]} : \texttt{Bool} \longrightarrow \texttt{Bool}}\ \text{t-tapp} \qquad \Gamma \vdash \texttt{true} : \texttt{Bool}}{\Gamma \vdash (\Lambda\alpha\ .\ \lambda x : \alpha\ .\ x)\ \texttt{[Bool]}\ \texttt{true} : \texttt{Bool}}\ \text{t-app}$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic doubling function: apply f twice to the argument x

$$\Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\ f\ (f\ x)$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

The polymorphic doubling function: apply f twice to the argument x

$$\Gamma \vdash (\quad \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\,(f\,x) \quad) : \forall \alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\texttt{Bool} = \forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

Functions of "two arguments" that choose either their first or second arguments.
We want them to work across all types
$\Longrightarrow$ the result type must be consistent with either choice
$\Longrightarrow$ the first and second argument types must match

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

$$\textbf{true} = (\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ ?\ )\ :\ \textbf{Bool}$$

$$\textbf{false} = (\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ ?\ )\ :\ \textbf{Bool}$$

The syntax of the type almost completely defines what the functions must be

$\forall \alpha\ .$ becomes $\Lambda \alpha\ .$ $\qquad$ $\alpha \longrightarrow \cdot$ becomes $\lambda x\ .$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall\alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

$$\textbf{true} = (\ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ ) : \textbf{Bool}$$

$$\textbf{false} = (\ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ ) : \textbf{Bool}$$

The only possibilities are returning the first or second argument

There are exactly two functions with this type signature

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\,[\tau] \qquad\qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\texttt{Bool} = \forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

$$\texttt{true} = (\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ ) : \texttt{Bool}$$

$$\texttt{false} = (\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ ) : \texttt{Bool}$$

$$\texttt{not} = (\qquad\qquad\qquad\qquad\qquad\qquad) : \texttt{Bool} \longrightarrow \texttt{Bool}$$

$\texttt{Bool} \longrightarrow \texttt{Bool}$ is shorthand for $(\forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha) \longrightarrow (\forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha)$

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall\alpha\ .\ \alpha \to \alpha \to \alpha$$

$$\textbf{true} = (\ \ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ \ ) : \textbf{Bool}$$

$$\textbf{false} = (\ \ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ \ ) : \textbf{Bool}$$

$$\textbf{not} = (\ \ \lambda b : \textbf{Bool}\ .\ \ \ ) : \textbf{Bool} \to \textbf{Bool}$$

Bool $\to$ **Bool** is shorthand for $(\forall\alpha\ .\ \alpha \to \alpha \to \alpha) \to (\forall\alpha\ .\ \alpha \to \alpha \to \alpha)$
t-abs tells us this must be an abstraction that takes an argument of type **Bool**

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

$$\textbf{true} = (\ \ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ \ \ ) : \textbf{Bool}$$

$$\textbf{false} = (\ \ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ \ \ ) : \textbf{Bool}$$

$$\textbf{not} = (\ \ \lambda b : \textbf{Bool}\ .\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ \qquad\qquad) : \textbf{Bool} \longrightarrow \textbf{Bool}$$

$\textbf{Bool} \longrightarrow \textbf{Bool}$ is shorthand for $(\forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha) \longrightarrow (\forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha)$
t-abs tells us this must be an abstraction that takes an argument of type **Bool**
t-app tells us the body must have type **Bool**, so it is also largely fixed (it could just be b)

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall \alpha\ .\ \alpha \longrightarrow \alpha \longrightarrow \alpha$$

$$\textbf{true} = (\ \ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ \ ) : \textbf{Bool}$$

$$\textbf{false} = (\ \ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ \ ) : \textbf{Bool}$$

$$\textbf{not} = (\ \ \lambda b : \textbf{Bool}\ .\ \Lambda \alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ \qquad\qquad) : \textbf{Bool} \longrightarrow \textbf{Bool}$$

This body must have type α

There are four choices here: there are four functions with the type **Bool** $\longrightarrow$ **Bool**

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Booleans in System F

$$\textbf{Bool} = \forall\alpha\ .\ \alpha \to \alpha \to \alpha$$

$$\textbf{true} = (\ \ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ x\ \ \ ) : \textbf{Bool}$$

$$\textbf{false} = (\ \ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ y\ \ \ ) : \textbf{Bool}$$

$$\textbf{not} = (\ \ \lambda b : \textbf{Bool}\ .\ \Lambda\alpha\ .\ \lambda x : \alpha\ .\ \lambda y : \alpha\ .\ b\,[\alpha]\ y\ x\ \ \ ) : \textbf{Bool} \to \textbf{Bool}$$

This is the interesting choice. The other choices:

x (always return **true**) y (always return **false**) $b\,[\alpha]\ x\ y$ (identity)

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

For any type of argument, apply a function zero or more times to the argument
$\Longrightarrow$ it must return the same type as the argument (e.g., for zero)
$\Longrightarrow$ the function must also take and return that same type

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall\alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$= (\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad) : \mathbf{Nat}$$

What expressions have type **Nat**?

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\begin{aligned} \textbf{Nat} &= \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha \\ &= (\quad \Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\qquad) : \textbf{Nat} \end{aligned}$$

As before, the type dictates much of the expression

However, there are infinitely many choices of body with type α

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$0 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\ x \quad) : \mathbf{Nat}$$

The simplest case is just x, which we call zero

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$0 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\ x \quad) : \mathbf{Nat}$$

$$1 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\ f\ x \quad) : \mathbf{Nat}$$

But we can also apply f to x, which we call one

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

$$0 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ x \quad) : \mathbf{Nat}$$

$$1 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\ x \quad) : \mathbf{Nat}$$

$$2 = (\quad \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\ (f\ x) \quad) : \mathbf{Nat}$$

Applying f to $f\ x$ also gives α, which we call two, and so on

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\textbf{Nat} = \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$\textbf{succ} = (\qquad\qquad) : \textbf{Nat} \longrightarrow \textbf{Nat}$$

We want a successor function that operates on **Nat**s

What expressions have type **Nat** $\longrightarrow$ **Nat**?

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau . e \mid \Lambda \alpha . e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha . \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha . e : \forall \alpha . \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha . \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha . (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat} . \qquad\qquad) : \mathbf{Nat} \longrightarrow \mathbf{Nat}$$

It must be a lambda term whose argument is type **Nat**

One correct choice of body is simply n, giving the identity function on Booleans

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat}\ .\ \Lambda \alpha\ .\ \lambda f : \alpha \longrightarrow \alpha\ .\ \lambda x : \alpha\ .\qquad\qquad) : \mathbf{Nat} \longrightarrow \mathbf{Nat}$$

The only other way to produce a **Nat** is to follow the "shape" of the type

This body must be an expression of type α

Unlike with Booleans, there are an infinite number of bodies of type α, such as x (constant zero), $f\ x$ (constant one), $f\ (f\ x)$ etc.

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat}\ .\ \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\,(n\,[\alpha]\,f\,x) \quad) : \mathbf{Nat} \to \mathbf{Nat}$$

To use n, we need to give it a type, a function on that type, and a value of that type.

One choice is $n\,[\alpha]\,f\,x$, which gives the identity function

Successor simply applies f to this

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda \alpha\ .\ e \mid e\,[\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda \alpha\ .\ e : \forall \alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall \alpha\ .\ \tau_1}{\Gamma \vdash e\,[\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall \alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat}\ .\ \Lambda \alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\,(n\,[\alpha]\,f\,x) \quad) : \mathbf{Nat} \to \mathbf{Nat}$$

$$\mathbf{plus} = (\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad) : \mathbf{Nat} \to \mathbf{Nat} \to \mathbf{Nat}$$

There are many functions that take two **Nat**s and produce a third.

Consider addition

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall\alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat}\ .\ \Lambda\alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\ (n\ [\alpha]\ f\ x) \quad) : \mathbf{Nat} \to \mathbf{Nat}$$

$$\mathbf{plus} = (\quad \lambda m : \mathbf{Nat}\ .\ \lambda n : \mathbf{Nat}\ .\ m\ [\mathbf{Nat}]\ \mathbf{succ}\ n \quad) : \mathbf{Nat} \to \mathbf{Nat} \to \mathbf{Nat}$$

One definition of the addition function applies **succ** m times to n

System F Typing Rules

$$e ::= x \mid e\ e \mid \lambda x : \tau\ .\ e \mid \Lambda\alpha\ .\ e \mid e\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall\alpha\ .\ \tau$$

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

$$\frac{\Gamma, \alpha \vdash e : \tau \quad \alpha \notin \Gamma}{\Gamma \vdash \Lambda\alpha\ .\ e : \forall\alpha\ .\ \tau}\ \text{t-tabs} \qquad \frac{\Gamma \vdash e : \forall\alpha\ .\ \tau_1}{\Gamma \vdash e\ [\tau_2] : \tau_1[\alpha := \tau_2]}\ \text{t-tapp}$$

Church Numerals in System F

$$\mathbf{Nat} = \forall\alpha\ .\ (\alpha \to \alpha) \to \alpha \to \alpha$$

$$\mathbf{succ} = (\quad \lambda n : \mathbf{Nat}\ .\ \Lambda\alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ f\ (n\ [\alpha]\ f\ x) \quad) : \mathbf{Nat} \to \mathbf{Nat}$$

$$\mathbf{plus} = (\quad \lambda m : \mathbf{Nat}\ .\ \lambda n : \mathbf{Nat}\ .\ m\ [\mathbf{Nat}]\ \mathbf{succ}\ n \quad) : \mathbf{Nat} \to \mathbf{Nat} \to \mathbf{Nat}$$

$$\mathbf{plus} = \lambda m : \mathbf{Nat}\ .\ \lambda n : \mathbf{Nat}\ .\ \Lambda\alpha\ .\ \lambda f : \alpha \to \alpha\ .\ \lambda x : \alpha\ .\ m\ [\alpha]\ f\ (n\ [\alpha]\ f\ x)$$

Another definition applies f to x n times, then applies f to this m times

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha \,.\, \tau$$

Variables and abstractions, both λ and Λ, are values

Values and applications, both term and type, are terms

Types built from type variabes, functions, and universally quantified types

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha \,.\, \tau$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{e-func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{e-arg} \qquad \frac{t[x := v] = t'}{(\lambda x : \tau \,.\, t)\ v \longrightarrow t'}\ \text{e-beta}$$

Call-by-value operational rules for applications like the simply-typed lambda calculus:

e-func: First reduce the “function” to a value

e-arg: Once the “function” is a value, reduce the argument to a value

e-beta: Apply a λ abstraction to a value argument by performing β reduction

Note that the type τ is discarded

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \to \tau \mid \forall \alpha \,.\, \tau$$

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2}\ \text{e-func} \qquad \frac{t \to t'}{v\ t \to v\ t'}\ \text{e-arg} \qquad \frac{t[x := v] = t'}{(\lambda x : \tau \,.\, t)\ v \to t'}\ \text{e-beta}$$

$$\frac{t_1 \to t_1'}{t_1\ [\tau] \to t_1'\ [\tau]}\ \text{e-tfunc} \qquad \frac{t[\alpha := \tau] = t'}{(\Lambda \alpha \,.\, t)\ [\tau] \to t'}\ \text{e-tbeta}$$

Additional rules for Λ abstractions and types:

e-tfunc: Reduce the "function" to a value before applying a type. Like e-func

e-tbeta: Apply a Λ abstraction to a type argument by performing β reduction. Like e-beta

Note that it is the type variable α that is being replaced with the type argument τ

No "e-targ" rule because types are always "values"

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha \,.\, \tau$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{e-func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{e-arg} \qquad \frac{t[x := v] = t'}{(\lambda x : \tau \,.\, t)\ v \longrightarrow t'}\ \text{e-beta}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ [\tau] \longrightarrow t_1'\ [\tau]}\ \text{e-tfunc} \qquad \frac{t[\alpha := \tau] = t'}{(\Lambda \alpha \,.\, t)\ [\tau] \longrightarrow t'}\ \text{e-tbeta}$$

Consider erasing types to transform a System F term to the lambda calculus:

$$\begin{aligned} \text{erase}(x) &= x & \text{erase}(\lambda x : \tau \,.\, t) &= \lambda x \,.\, \text{erase}(t) \\ \text{erase}(t_1\ t_2) &= \text{erase}(t_1)\ \text{erase}(t_2) & \text{erase}(\Lambda \alpha \,.\, t) &= \text{erase}(t) \\ \text{erase}(t\ [\tau]) &= \text{erase}(t) \end{aligned}$$

More efficient evaluation: types are not consulted during evaluation

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha \,.\, \tau$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{e-func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{e-arg} \qquad \frac{t[x := v] = t'}{(\lambda x : \tau \,.\, t)\ v \longrightarrow t'}\ \text{e-beta}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ [\tau] \longrightarrow t_1'\ [\tau]}\ \text{e-tfunc} \qquad \frac{t[\alpha := \tau] = t'}{(\Lambda \alpha \,.\, t)\ [\tau] \longrightarrow t'}\ \text{e-tbeta}$$

Consider erasing types to transform a System F term to the lambda calculus:

$$\begin{aligned}
\text{erase}(x) &= x & \text{erase}(\lambda x : \tau \,.\, t) &= \lambda x \,.\, \text{erase}(t) \\
\text{erase}(t_1\ t_2) &= \text{erase}(t_1)\ \text{erase}(t_2) & \text{erase}(\Lambda \alpha \,.\, t) &= \text{erase}(t) \\
\text{erase}(t\ [\tau]) &= \text{erase}(t)
\end{aligned}$$

The type reconstruction problem: given a Lambda Calculus term m, is there a well-typed System F term t such that $\text{erase}(t) = m$?

System F Call-By-Value Evaluation Rules

$$v ::= x \mid \lambda x : \tau \,.\, t \mid \Lambda \alpha \,.\, t \qquad t ::= v \mid t\ t \mid t\ [\tau] \qquad \tau ::= \alpha \mid \tau \longrightarrow \tau \mid \forall \alpha \,.\, \tau$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{e-func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{e-arg} \qquad \frac{t[x := v] = t'}{(\lambda x : \tau \,.\, t)\ v \longrightarrow t'}\ \text{e-beta}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ [\tau] \longrightarrow t_1'\ [\tau]}\ \text{e-tfunc} \qquad \frac{t[\alpha := \tau] = t'}{(\Lambda \alpha \,.\, t)\ [\tau] \longrightarrow t'}\ \text{e-tbeta}$$

Consider erasing types to transform a System F term to the lambda calculus:

$$\begin{aligned} \text{erase}(x) &= x & \text{erase}(\lambda x : \tau \,.\, t) &= \lambda x \,.\, \text{erase}(t) \\ \text{erase}(t_1\ t_2) &= \text{erase}(t_1)\ \text{erase}(t_2) & \text{erase}(\Lambda \alpha \,.\, t) &= \text{erase}(t) \\ \text{erase}(t\ [\tau]) &= \text{erase}(t) \end{aligned}$$

Unfortunately not: [Wells, 1994] whether there exists a well-typed term t such that $\text{erase}(t) = m$ is undecidable

Hindley-Milner is a real “sweet spot”

Theorem

Preservation: *If* $\Gamma \vdash e : \tau$ *and* $e \longrightarrow e'$ *then* $\Gamma \vdash e' : \tau$

Theorem

Preservation: *If* $\Gamma \vdash e : \tau$ *and* $e \longrightarrow e'$ *then* $\Gamma \vdash e' : \tau$

Theorem

Progress: *If* e *is a closed, well-typed term,* e *is a value or there is some* e' *such that* $e \longrightarrow e'$

Theorem

Preservation: *If* $\Gamma \vdash e : \tau$ *and* $e \longrightarrow e'$ *then* $\Gamma \vdash e' : \tau$

Theorem

Progress: *If* e *is a closed, well-typed term,* e *is a value or there is some* e' *such that* $e \longrightarrow e'$

$\Longrightarrow$ System F is type-safe

Theorem

Preservation: *If* $\Gamma \vdash e : \tau$ *and* $e \longrightarrow e'$ *then* $\Gamma \vdash e' : \tau$

Theorem

Progress: *If* e *is a closed, well-typed term,* e *is a value or there is some* e' *such that* $e \longrightarrow e'$

$\Longrightarrow$ System F is type-safe

Theorem

Normalization: *Evaluating a well-typed term always leads to a value*

Wait, what?

$\lambda x\,.\,x\,x$ is OK: $(\lambda x : (\forall \alpha\,.\,\alpha \longrightarrow \alpha)\,.\,x\,[\forall \alpha\,.\,\alpha \longrightarrow \alpha]\,x) : (\forall \alpha\,.\,\alpha \longrightarrow \alpha) \longrightarrow (\forall \alpha\,.\,\alpha \longrightarrow \alpha)$

But $(\lambda x\,.\,x\,x)\,(\lambda x\,.\,x\,x)$ is not well-typed in System F