A History of Subtyping

Benjamin C. Pierce
University of Pennsylvania

PLMW, August 2023, Seattle

~~A History of Subtyping~~

Benjamin C. Pierce
University of Pennsylvania

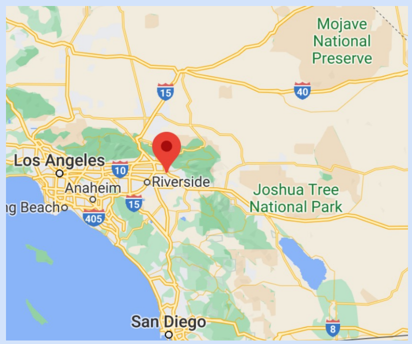
PLMW, August 2023, Seattle

What Does Subtyping Mean?

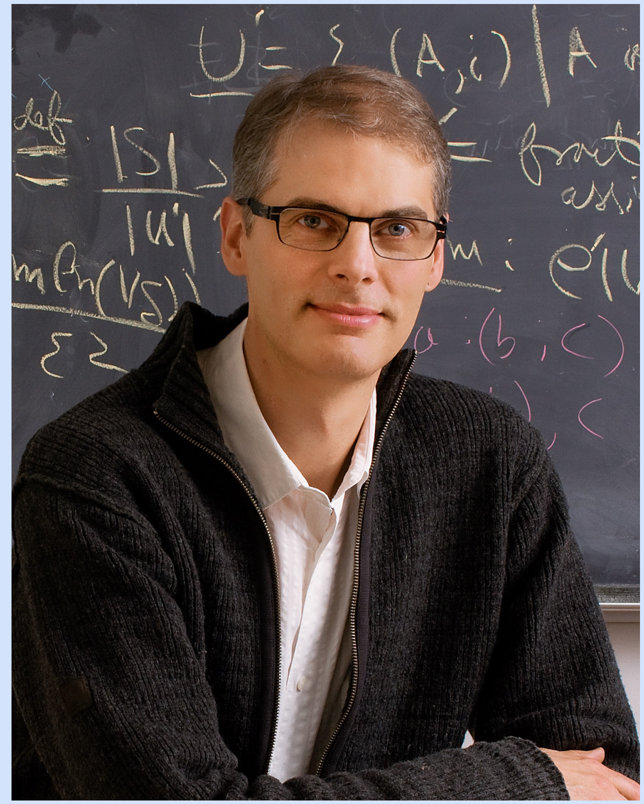
Benjamin C. Pierce
University of Pennsylvania

PLMW, August 2023, Seattle

- Grew up in Redlands, CA
  - About halfway between Los Angeles and Palm Springs



- PhD from CMU
  - Advised by Bob Harper and John Reynolds
- Postdocs at Edinburgh, INRIA, Cambridge
  - With Robin Milner, Didier Rémy
- Taught at Indiana University for two years
- At Penn since 1998



Me

1963 -

I like...

writing

music

contact improv

Kids :-)

What about you?

Ask the people on both sides of you…

- Name?
- Hometown?
- Favorite kind of music?
- Favorite language with subtyping?

What this talk is about

- Some basic stuff about typing and subtyping
  - (that may be familiar)
- Some other basic stuff
  - (that may be less familiar)
- Some history
- Some people

Please interrupt me!

What Does Subtyping Mean?

~~What Does Subtyping Mean?~~

What Does Typing Mean?

What Does Typing Mean?

Let's return to the source…

The Lambda Calculus

Syntax

expressions

$$e \quad ::= \quad c \mid x \mid \lambda x.e_1 \mid e_1\ e_2$$

constants

variables

function abstractions

function applications

Operational Semantics

$$(\lambda x.\ e_1)\ e_2 \longrightarrow [e_2/x]e_1 \qquad \text{(R-BETA)}$$

$$\frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2} \qquad \text{(R-APP1)}$$

$$\frac{e_2 \longrightarrow e_2'}{e_1\ e_2 \longrightarrow e_1\ e_2'} \qquad \text{(R-APP2)}$$

$$\frac{e_1 \longrightarrow e_1'}{\lambda x.\ e_1 \longrightarrow \lambda x.\ e_1'} \qquad \text{(R-ABS)}$$

Example

$$(\lambda x.\ \lambda y.\ x)\ s\ t \longrightarrow (\lambda y.\ s)\ t \longrightarrow s$$

Formally...

$$\dfrac{\dfrac{}{(\lambda x.\ \lambda y.\ x)\ s \longrightarrow \lambda y.\ s}\ (\textsc{R-Beta})}{(\lambda x.\ \lambda y.\ x)\ s\ t \longrightarrow (\lambda y.\ s)\ t}\ (\textsc{R-App1})$$

Aside: Reduction Strategies

Most programming languages restrict this "full beta-reduction" to a deterministic *function*.

- Call by name
- Call by value
- Lazy
- Etc.

These distinctions are not needed for this talk.

Known for …

- the lambda calculus
- the Church–Turing thesis
  - … that *every effectively calculable function is a computable function*
- the undecidability of first-order logic
- (and much more!)

"With his doctoral student Alan Turing, Church is considered one of the founders of computer science." [Wikipedia]

Alonzo Church

1903-1995

Alonzo Church

Biography MathSciNet

Ph.D. Princeton University 1927

Dissertation: *Alternatives to Zermelo's Assumption*

Mathematics Subject Classification: 06—Order, lattices, ordered algebraic structures

Advisor: Oswald Veblen

Students:
Click here to see the students listed in chronological order.

From the Mathematics Geneology project…

Name	School	Year	Descendants
Anderson, C. Anthony	University of California, Los Angeles	1977	
Andrews, Peter	Princeton University	1964	94
Arbab, Bijan	University of California, Los Angeles	1988	
Barnard, George	Princeton University	1936	466
Bennett, James	Princeton University	1962	
Boone, William	Princeton University	1952	30
Bustamente-Llaca, Enrique	Princeton University	1944	
Chapin, Edward	Princeton University	1970	1
Collins, Donald	Princeton University	1967	16
Daigneault, Aubert	Princeton University	1959	
Davis, Martin	Princeton University	1950	66
Easton, William	Princeton University	1964	
Foster, Alfred	Princeton University	1930	307
Guard, James	Princeton University	1961	2
Henkin, Leon	Princeton University	1947	86
Hensel, Gustav	Princeton University	1963	
Kemeny, John	Princeton University	1949	84
Kleene, Stephen	Princeton University	1934	1465
Kochen, Simon	Princeton University	1959	38
L'Abbé, Maurice	Princeton University	1951	
Malitz, Isaac (Richard)	University of California, Los Angeles	1976	
Mar, Gary	University of California, Los Angeles	1985	1
Massey, Gerald	Princeton University	1964	230
Rabin, Michael	Princeton University	1957	459
Rescher, Nicholas	Princeton University	1951	70
Richter, Wayne	Princeton University	1963	203
Ritchie, Robert	Princeton University	1960	51
Robbin, Joel	Princeton University	1965	11
Rogers, Jr., Hartley	Princeton University	1952	843
Rosser, J. Barkley	Princeton University	1934	1168
Routley, Richard	Princeton University	1981	1
Scott, Dana	Princeton University	1958	623
Shapiro, Norman	Princeton University	1955	
Smullyan, Raymond	Princeton University	1959	12
Turing, Alan	Princeton University	1938	230
Winder, Robert	Princeton University	1962	

According to our current on-line database, Alonzo Church has 36 students and 6307 descendants.
We welcome any additional information.

Alonzo Church

Biography MathSciNet

Ph.D. Princeton University 1927

Dissertation: *Alternatives to Zermelo's Assumption*

Mathematics Subject Classification: 06—Order, lattices, ordered algebraic structures

Advisor: Oswald Veblen

Students:
Click here to see the students listed in chronological order.

Name	School	Year	Desc
Anderson, C. Anthony	University of California, Los Angeles	1977	
Andrews, Peter	Princeton University	1964	
Arbab, Bijan	University of California, Los Angeles	1988	
Barnard, George	Princeton University	1936	
Bennett, James	Princeton University	1962	
Boone, William	Princeton University	1952	30
Bustamente-Llaca, Enrique	Princeton University	1944	
Chapin, Edward	Princeton University	1970	
Collins, Donald	Princeton University	1967	
Daigneault, Aubert	Princeton University	1959	
Davis, Martin	Princeton University	1950	66
Easton, William	Princeton University	1964	
Foster, Alfred	Princeton University	1930	307
Guard, James	Princeton University	1961	2
Henkin, Leon	Princeton University	1947	86

From the Mathematics Geneology project…

Hensel, Gustav	Princeton University	1963	
Kemeny, John	Princeton University	1949	84
Massey, Gerald	Princeton	34	1465
Rabin, Michael	Princeton	59	38
Rescher, Nicholas	Princeton	51	
Rosser, J. Barkley	Princeton University	76	
Routley, Richard	Princeton University	85	1
Scott, Dana	Princeton University	1964	230
Shapiro, Norman	Princeton University		459
Smullyan, Raymond	Princeton University		70
Turing, Alan	Princeton University		203
Winder, Robert	Princeton University		51
			11
			843
Rosser, J. Barkley	Princeton University	1934	1168
Winder, Robert	Princeton University	1962	

According to our current on-line database, Alonzo Church has 36 students and 6307 descendants.
We welcome any additional information.

Adding Records

construction

projection

$$e \quad ::= \quad \ldots \mid \{l_1{=}e_1, \ldots, l_n{=}e_n\} \mid e.l$$

$$\frac{e_i \longrightarrow e_i'}{\{\ldots,\ l_i{=}e_i\,, \ldots\} \longrightarrow \{\ldots,\ l_i{=}e_i'\,, \ldots\}} \quad \text{(R-RCD)}$$

$$\frac{}{\{\ldots,\ l_i{=}e_i\,, \ldots\}.l_i \longrightarrow e_i} \quad \text{(R-RCDPROJ)}$$

Examples

$\lambda x{:}\text{String}.\ x$

3.8 0.0

$\lambda x.\ x\ x$

$\lambda x{:}\mathbb{R}.\ x$

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

{n='Bob'}

$(\lambda x{:}\text{Student}.\ x.\text{n})$ {n='Alice', g=3.8}

'Alice'

'Bob'

'Alice' 3.8

$(\lambda x{:}\text{Student}.\ \lambda y{:}\text{Person}.\ x)$ {n='Bob'} {n='Alice', g=3.8}

$(\lambda y{:}\text{Person}.$ {n='Alice', g=3.8}$)$ {n='Bob'}

{n='Alice', g=3.8}

Examples

functions

$\lambda x{:}\mathsf{String}.\ x$

3.8 0.0

$\lambda x.\ x\ x$

$\lambda x{:}\mathbb{R}.\ x$

constants

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

{n='Bob'}

'Alice'

'Bob'

$(\lambda x{:}\mathsf{Student}.\ x.\mathsf{n})$ {n='Alice', g=3.8}

'Alice' 3.8

compound expressions

records

$(\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)$ {n='Bob'} {n='Alice', g=3.8}

$(\lambda y{:}\mathsf{Person}.$ {n='Alice', g=3.8}$)$ {n='Bob'}

{n='Alice', g=3.8}

The Simply Typed Lambda-Calculus

Syntax

base types

function types

record types

$$
\begin{aligned}
T &::= B \mid T_1 \to T_2 \mid \{l_1{:}T_1, \ldots, l_n{:}T_n\} \\
e &::= c \mid x \mid \lambda x{:}T.\ e_1 \mid e_1\ e_2 \mid \{l_1{=}e_1, \ldots, l_n{=}e_n\} \mid e.l \\
\Gamma &::= \cdot \mid \Gamma_1,\ x{:}T
\end{aligned}
$$

empty context

extended context

Example

$$
\begin{aligned}
\mathsf{Person} &:= \{\mathsf{n}\colon \mathsf{String}\} \\
\mathsf{Student} &:= \{\mathsf{n}\colon \mathsf{String},\ \mathsf{g}\colon \mathsf{Real}\}
\end{aligned}
$$

$$
\begin{array}{lcllcl}
\mathsf{bob} & := & \{\mathsf{n}='\mathsf{Bob}'\} & : & \mathsf{Person} \\
\mathsf{alice} & := & \{\mathsf{n}='\mathsf{Alice}',\ \mathsf{g}=3.8\} & : & \mathsf{Student}
\end{array}
$$

Typing

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x \in T} \quad \text{(T-Var)}$$

$$\frac{\Gamma,\ x{:}T_1 \vdash e_1 \in T_2}{\Gamma \vdash \lambda x{:}T_1.\ e_1 \in T_1 \to T_2} \quad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash e_1 \in T_1 \to T_2 \qquad \Gamma \vdash e_2 \in T_1}{\Gamma \vdash e_1\ e_2 \in T_2} \quad \text{(T-App)}$$

$$\frac{\Gamma \vdash e_1 \in T_1 \quad \ldots \quad \Gamma \vdash e_n \in T_n}{\Gamma \vdash \{l_1{=}e_1,\ \ldots,\ l_n{=}e_n\} \in \{l_1{:}T_1,\ \ldots,\ l_n{:}T_n\}} \quad \text{(T-Rcd)}$$

$$\frac{\Gamma \vdash e \in \{l_1{:}T_1,\ \ldots,\ l_n{:}T_n\}}{\Gamma \vdash e.l_i \in T_i} \quad \text{(T-Proj)}$$

$$\frac{\mathit{typeOfConst}(c) = C}{\Gamma \vdash c \in C} \quad \text{(T-Const)}$$

Typing Derivations

subderivation for S

subderivation for T

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        x{:}\,S \in (x{:}\,S,\ y{:}\,T)
      }{
        x{:}\,S,\ y{:}\,T \vdash x \in S
      }\ \text{(T-Var)}
    }{
      x{:}\,S \vdash \lambda y{:}T.\ x \in T \to S
    }\ \text{(T-Abs)}
  }{
    \cdot \vdash \lambda x{:}S.\ \lambda y{:}T.\ x \in S \to T \to S
  }\ \text{(T-Abs)}
  \qquad
  \begin{array}{c} \Delta_1 \\ \cdots\cdots\cdots \\ \cdot \vdash s \in S \end{array}
}{
  \cdot \vdash (\lambda x{:}S.\ \lambda y{:}T.\ x)\ s \in T \to S
}\ \text{(T-App)}
\qquad
\dfrac{}{}
\begin{array}{c} \Delta_2 \\ \cdots\cdots\cdots \\ \cdot \vdash t \in T \end{array}
$$

$$
\dfrac{\cdot \vdash (\lambda x{:}S.\ \lambda y{:}T.\ x)\ s \in T \to S \qquad \cdot \vdash t \in T}{\cdot \vdash (\lambda x{:}S.\ \lambda y{:}T.\ x)\ s\ t \in S}\ \text{(T-App)}
$$

Correctness

Theorem (Preservation):

If $\Gamma \vdash e \in U$ and $e \longrightarrow e'$,
then $\Gamma \vdash e' \in U$.

In particular:

If $\Gamma \vdash (\lambda x{:}T.\ e_1)\ e_2 \in U$,
then $\Gamma \vdash [e_2/x]e_1 \in U$.

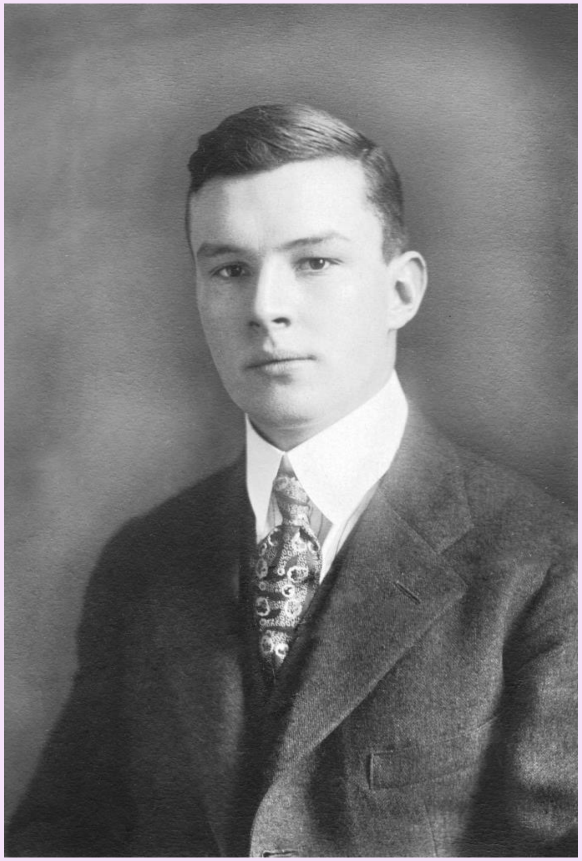
What does typing “mean”?

Known (in PL) especially for:

- the **Curry-Howard Correspondence** between the fundamental structures found in logic and in computation
- And, of course, the “currying” operation on multi-argument functions

S x T -> U ~ S -> T -> U

Also: How many people have three PLs named after them??



Haskell B. Curry

1900 - 1982

“Church style” vs. “Curry style”

“There are two versions of type assignment in the λ-calculus:

- **Church-style**, in which the type of each variable is fixed, and
- **Curry-style** (also called “domain free”), in which it is not.

As an example, in Church-style typing, $\lambda x{:}A.x$ is the identity function on type A, and it has type $A \to A$ but not $B \to B$ for a type B different from A.

In Curry-style typing, $\lambda x.x$ is a general identity function with type $C \to C$ for every type C.”

Bridging Curry and Church’s typing style, Kamareddin et al, 2016

But the distinction goes deeper…

I.e., this is not “just a matter of type inference”

“Extrinsic” View

“Extrinsic” View

Terms come first

λx:String. x

3.8 0.0

$\lambda x.\ x\ x$

{n='Bob'}

$\lambda x{:}\mathbb{R}.\ x$

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

(λx:Student. x.n) {n='Alice', g=3.8}

'Alice'

'Bob'

'Alice' 3.8

(λx:Student. λy:Person. x) {n='Bob'} {n='Alice', g=3.8}

(λy:Person. {n='Alice', g=3.8}) {n='Bob'}

{n='Alice', g=3.8}

“Extrinsic” View

Terms come first
Then reduction

$\lambda x{:}\mathsf{String}.\ x$

3.8 0.0

$\lambda x.\ x\ x$

{n='Bob'}

$\lambda x{:}\mathbb{R}.\ x$

$(\lambda x.\ x\ x)\ (\lambda x.\ x\ x)$

'Alice'

$(\lambda x{:}\mathsf{Student}.\ x.\mathsf{n})$ {n='Alice', g=3.8}

'Bob'

'Alice' 3.8

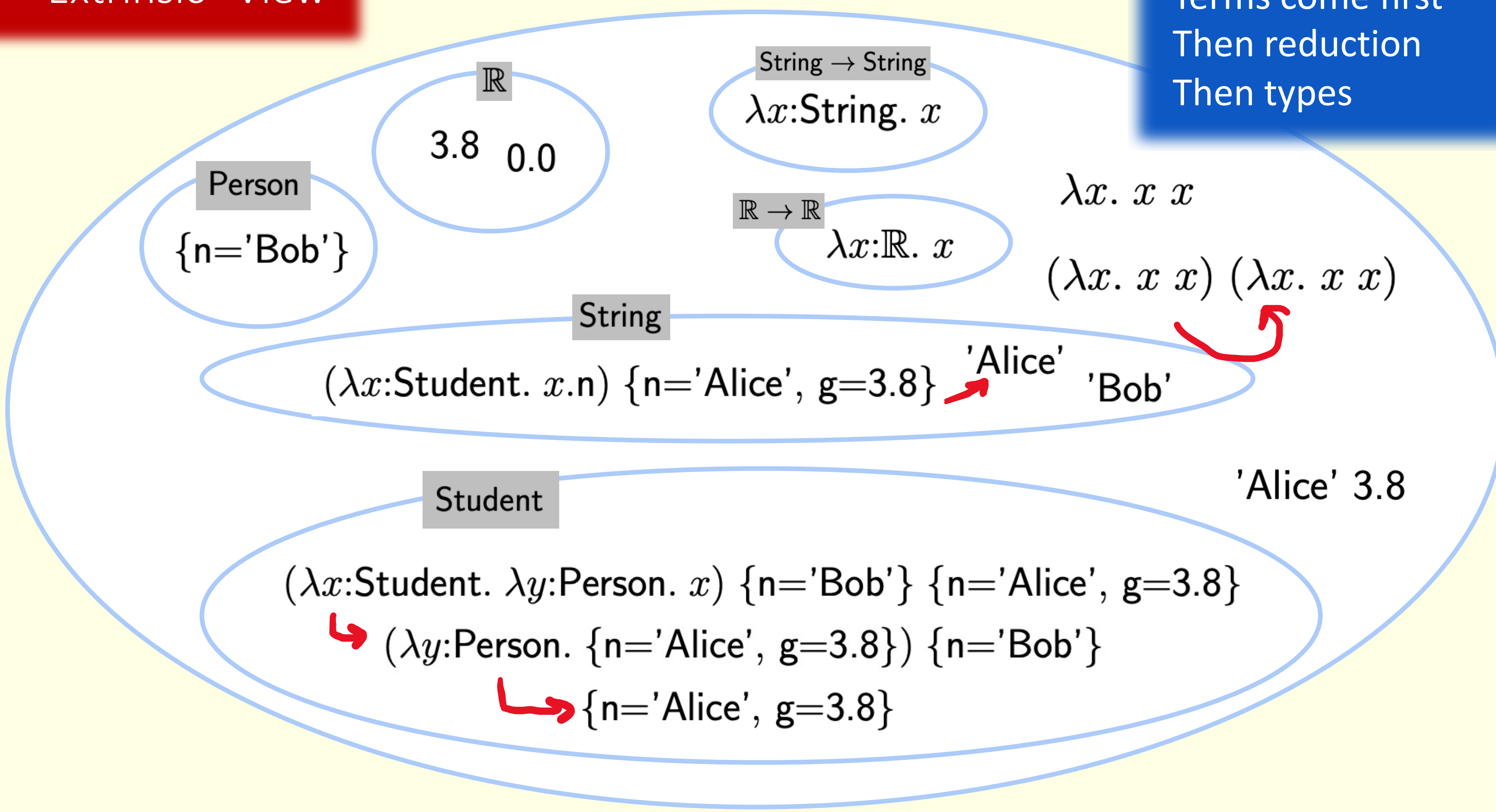
$(\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)$ {n='Bob'} {n='Alice', g=3.8}

$(\lambda y{:}\mathsf{Person}.$ {n='Alice', g=3.8}$)$ {n='Bob'}

{n='Alice', g=3.8}

“Extrinsic” View

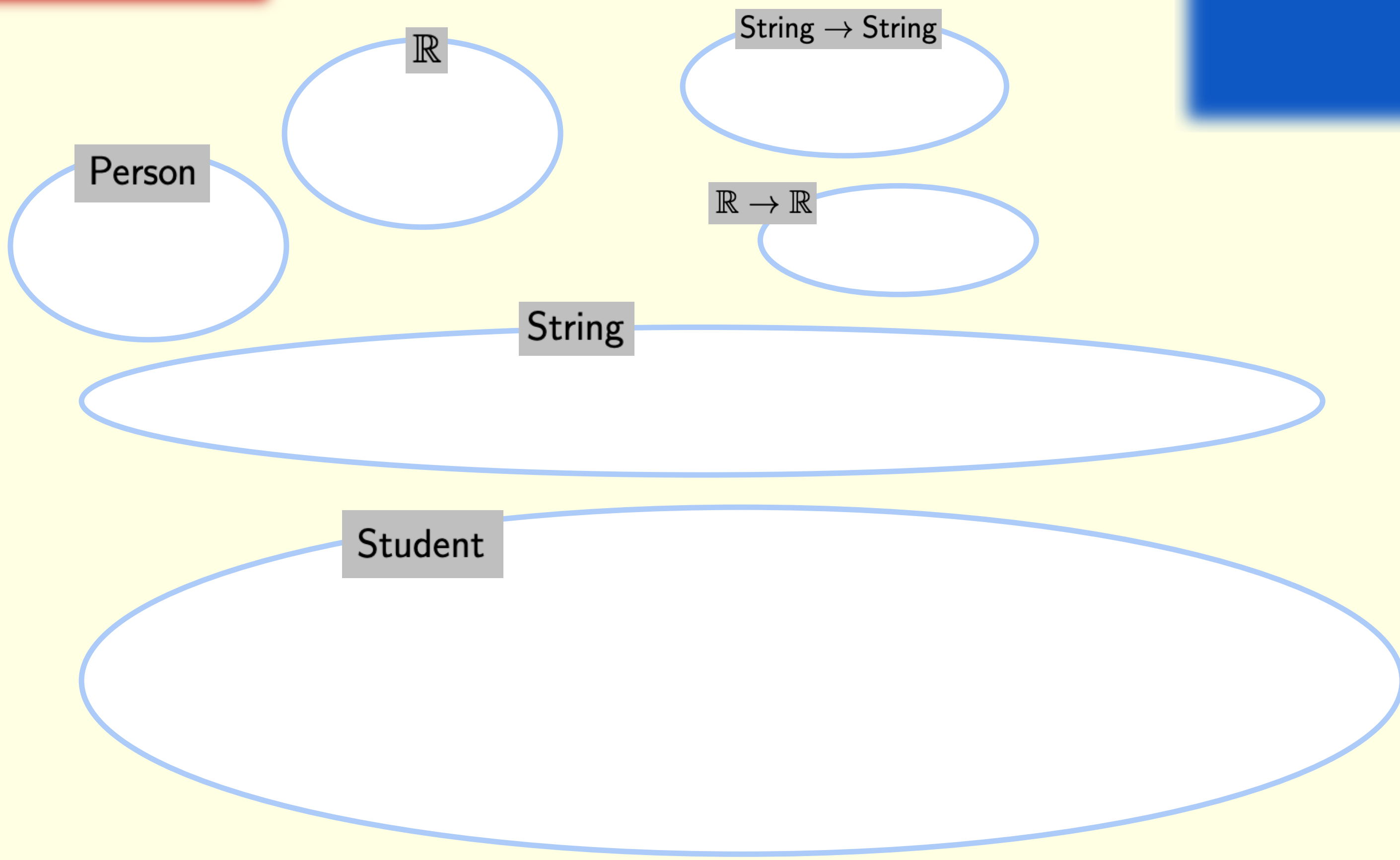
Terms come first
Then reduction
Then types



“Intrinsic” View

“Intrinsic” View

Types come first



“Intrinsic” View

Types come first
Then (typed) terms

$\mathbb{R}$

$\cdot \vdash 3.8 \in \mathbb{R}$

$\cdot \vdash 0.0 \in \mathbb{R}$

String $\rightarrow$ String

$\lambda x{:}\text{String}.\ x$

Person

$\cdot \vdash \{\text{n='Bob'}\} \in \text{Person}$

$\mathbb{R} \rightarrow \mathbb{R}$

$\lambda x{:}\mathbb{R}.\ x$

String

$\cdot \vdash (\lambda x{:}\text{Student}.\ x.\text{n})\ \{\text{n='Alice'},\ \text{g=3.8}\} \in \text{String}$

$\cdot \vdash \text{'Alice'} \in \text{String}$

$\cdot \vdash \text{'Bob'} \in \text{String}$

Student

$\cdot \vdash (\lambda x{:}\text{Student}.\ \lambda y{:}\text{Person}.\ x)\ \{\text{n='Bob'}\}\ \{\text{n='Alice'},\ \text{g=3.8}\} \in \text{Student}$

$\cdot \vdash (\lambda y{:}\text{Person}.\ \{\text{n='Alice'},\ \text{g=3.8}\})\ \{\text{n='Bob'}\} \in \text{Student}$

$\cdot \vdash \{\text{n='Alice'},\ \text{g=3.8}\} \in \text{Student}$

"Intrinsic" View

Types come first
Then (typed) terms
Then reduction

$\mathbb{R}$

$\cdot \vdash 3.8 \in \mathbb{R}$

$\cdot \vdash 0.0 \in \mathbb{R}$

$\mathsf{String} \to \mathsf{String}$

$\lambda x{:}\mathsf{String}.\ x$

Person

$\cdot \vdash \{\mathsf{n}='\mathsf{Bob}'\} \in \mathsf{Person}$

$\mathbb{R} \to \mathbb{R}$

$\lambda x{:}\mathbb{R}.\ x$

String

$\cdot \vdash (\lambda x{:}\mathsf{Student}.\ x.\mathsf{n})\ \{\mathsf{n}='\mathsf{Alice}',\ \mathsf{g}=3.8\} \in \mathsf{String}$

$\cdot \vdash '\mathsf{Alice}' \in \mathsf{String}$

$\cdot \vdash '\mathsf{Bob}' \in \mathsf{String}$

Student

$\cdot \vdash (\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)\ \{\mathsf{n}='\mathsf{Bob}'\}\ \{\mathsf{n}='\mathsf{Alice}',\ \mathsf{g}=3.8\} \in \mathsf{Student}$

$\hookrightarrow \cdot \vdash (\lambda y{:}\mathsf{Person}.\ \{\mathsf{n}='\mathsf{Alice}',\ \mathsf{g}=3.8\})\ \{\mathsf{n}='\mathsf{Bob}'\} \in \mathsf{Student}$

$\hookrightarrow \cdot \vdash \{\mathsf{n}='\mathsf{Alice}',\ \mathsf{g}=3.8\} \in \mathsf{Student}$

“Intrinsic” View

Types come first
Then (typed) terms

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x{:}\,\mathsf{Student} \in (x{:}\,\mathsf{Student},\ y{:}\,\mathsf{Person})}{x{:}\,\mathsf{Student},\ y{:}\,\mathsf{Person} \vdash x \in \mathsf{Student}}\,(\text{T-Var})}{x{:}\,\mathsf{Student} \vdash \lambda y{:}\mathsf{Person}.\ x \in \mathsf{Person} \to \mathsf{Student}}\,(\text{T-Abs})}{\cdot \vdash \lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x \in \mathsf{Student} \to \mathsf{Person} \to \mathsf{Student}}\,(\text{T-Abs}) \quad \dfrac{\Delta_1}{\cdot \vdash \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\} \in \mathsf{Student}}}{\cdot \vdash (\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)\ \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\} \in \mathsf{Person} \to \mathsf{Student}}\,(\text{T-App}) \quad \dfrac{\Delta_2}{\cdot \vdash \{\mathsf{n}{=}\text{'Bob'}\} \in \mathsf{Person}}}{\cdot \vdash (\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)\ \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\}\ \{\mathsf{n}{=}\text{'Bob'}\} \in \mathsf{Student}}\,(\text{T-App})$$

$\cdot \vdash (\lambda x{:}\mathsf{Student}.\ \lambda y{:}\mathsf{Person}.\ x)\ \{\mathsf{n}{=}\text{'Bob'}\}\ \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\} \in \mathsf{Student}$

$\hookrightarrow \cdot \vdash (\lambda y{:}\mathsf{Person}.\ \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\})\ \{\mathsf{n}{=}\text{'Bob'}\} \in \mathsf{Student}$

$\hookrightarrow \cdot \vdash \{\mathsf{n}{=}\text{'Alice'},\ \mathsf{g}{=}3.8\} \in \mathsf{Student}$

Types come first

Then (typed) terms

Then reduction

Types come first
Then typing derivations
Then reduction

Reduction on typing derivations??

Sure!

Reduction on Derivations

$$
\dfrac{
  \dfrac{
    \dfrac{\Delta_1}{\Gamma,\, x{:}T \vdash e_1 \in U}
  }{
    \Gamma \vdash \lambda x{:}T.\ e_1 \in T \to U
  }\ (\textsc{T-Abs})
  \qquad
  \dfrac{\Delta_2}{\Gamma \vdash e_2 \in T}
}{
  \Gamma \vdash (\lambda x{:}T.\ e_1)\ e_2 \in U
}\ (\textsc{T-App})
$$

Reduction on Derivations

$$
\dfrac{\dfrac{\begin{array}{c}\Delta_1\\ \cdots\cdots\cdots\\ \Gamma,\, x{:}T \vdash e_1 \in U\end{array}}{\Gamma \vdash \lambda x{:}T.\ e_1 \in T \to U}\ (\text{T-Abs}) \qquad \begin{array}{c}\Delta_2\\ \cdots\cdots\cdots\\ \Gamma \vdash e_2 \in T\end{array}}{\Gamma \vdash (\lambda x{:}T.\ e_1)\ e_2 \in U}\ (\text{T-App})
$$

$$
\longrightarrow \qquad \begin{array}{c}[\Delta_2/x]\Delta_1\\ \cdots\cdots\cdots\\ \Gamma \vdash [e_2/x]e_1 \in U\end{array}
$$

i.e., in Δ_1 replace every leaf where the T-Var rule is used to look up x with a copy of Δ_2

For example…

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{x{:}\,S \in (x{:}\,S,\ y{:}\,T)}{x{:}\,S,\ y{:}\,T \vdash x \in S}\,\text{(T-Var)}
}{x{:}\,S \vdash \lambda y{:}T.\ x \in T \to S}\,\text{(T-Abs)}
}{\cdot \vdash \lambda x{:}S.\ \lambda y{:}T.\ x \in S \to T \to S}\,\text{(T-Abs)}
\qquad
\dfrac{\Delta_1}{\cdot \vdash s \in S}
}{\cdot \vdash (\lambda x{:}S.\ \lambda y{:}T.\ x)\ s \in T \to S}\,\text{(T-App)}
\qquad
\dfrac{\Delta_2}{\cdot \vdash t \in T}
}{\cdot \vdash (\lambda x{:}S.\ \lambda y{:}T.\ x)\ s\ t \in S}\,\text{(T-App)}
$$

$$\longrightarrow$$

$$
\dfrac{
\dfrac{
\dfrac{\Delta_1}{y{:}\,T \vdash s \in S}
}{\cdot \vdash \lambda y{:}T.\ s \in T \to S}\,\text{(T-Abs)}
\qquad
\dfrac{\Delta_2}{\cdot \vdash t \in T}
}{\cdot \vdash (\lambda y{:}T.\ s)\ t \in S}\,\text{(T-App)}
$$

Subtyping

Motivation

A perfectly reasonable program that is
not typeable in the STLC…

$$(\lambda x{:}\mathsf{Person}.\ x.\mathsf{n})\ \mathsf{alice}$$

:-(

Inventors of the Simula and Simula-67 languages

Simula-67 was the first language to incorporate subtyping

- (The underlying idea was inspired by Tony Hoare)

Ole-Johan Dahl
Kristin Nygaard

1931 – 2002
1926 - 2002

In some research teams a new idea is treated with loving care: "How interesting!", "Beautiful!". This was not the case in the SIMULA development. When one of us announced that he had a new idea, the other would brighten up and do his best to kill it off. Assuming that the person who got the idea is willing to fight, this is a far better mode of work than the mode of mutual admiration. We think it was useful for us, and we succeeded in discarding a very large number of proposals.

Long career in famous research labs (Bell Labs, DEC SRC, Microsoft Cambridge); currently at Oxford

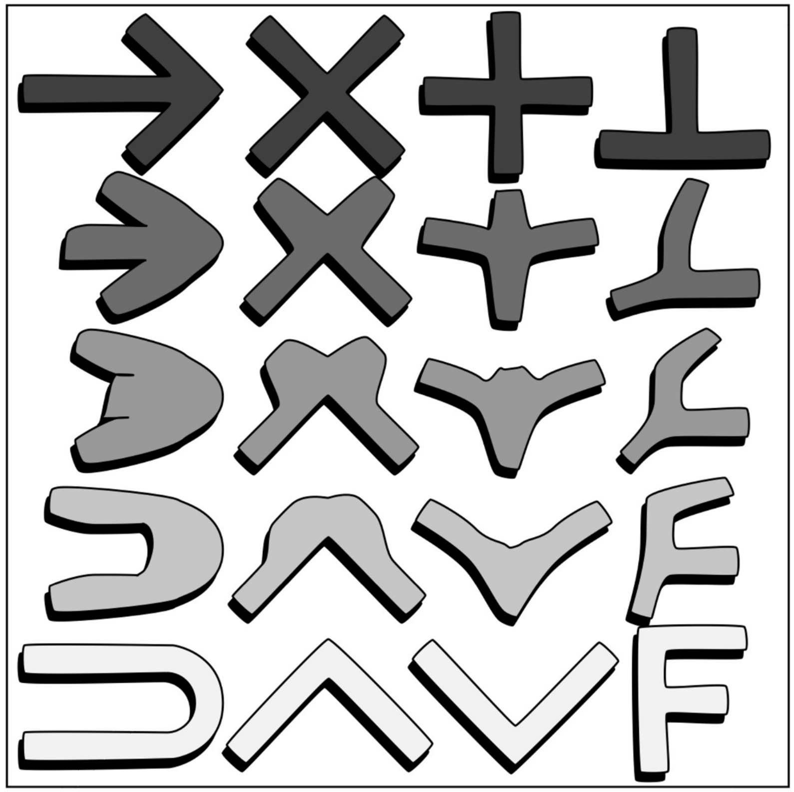
Many contributions to PL (and systems biology!)

- “Typeful programming”
- Bounded quantification (System $F_{<:}$)
- Record calculi
- Mobile Ambients
- A Theory of Objects (with Abadi)

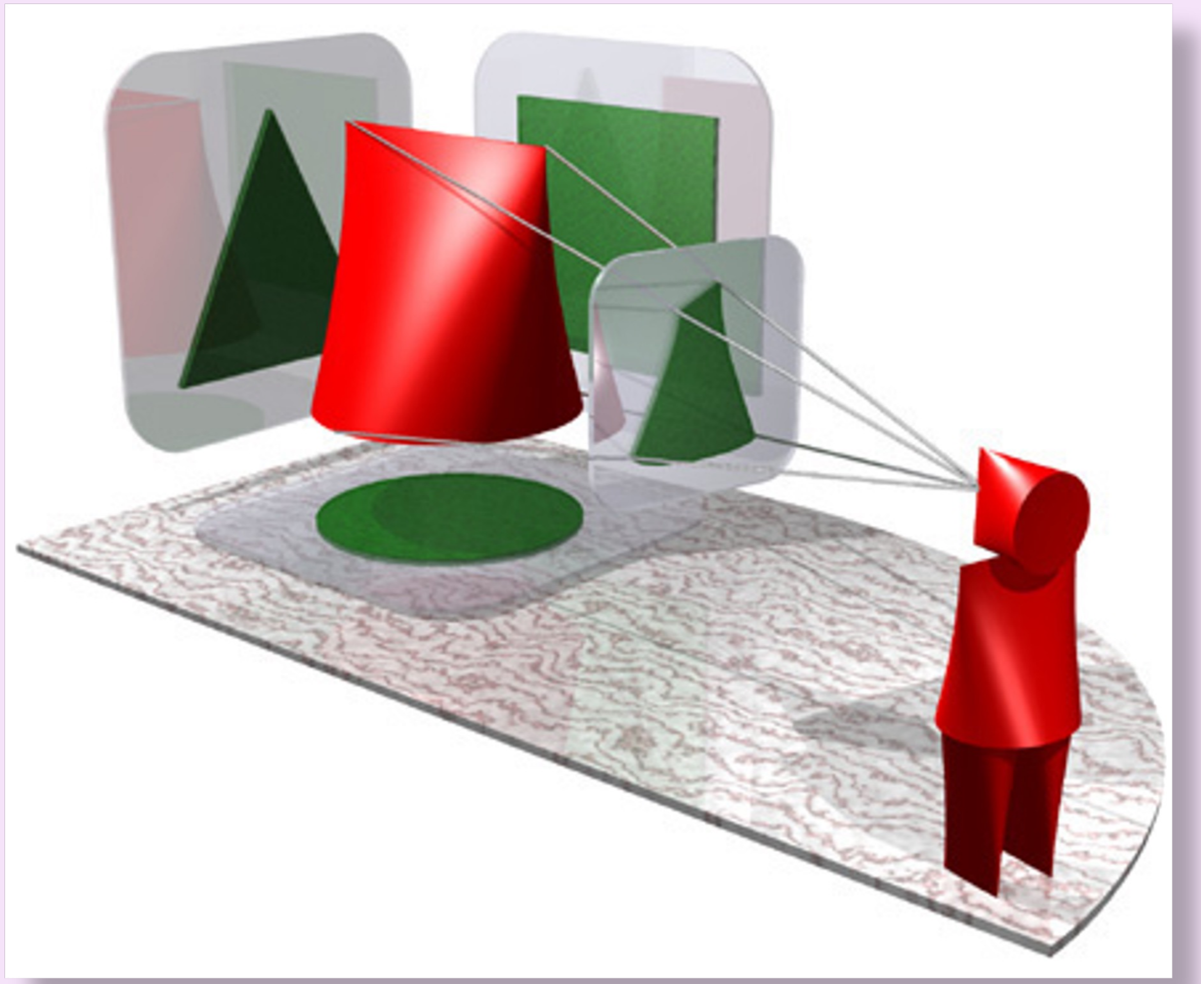
Winner of the Dahl-Nygaard prize in 2007 (among many other awards)

Luca Cardelli

1954? -



LC'90 *The Curry-Howard homeomorphism*




à
J.-Y. GIRARD
l'inventeur du
Système F
<:
rules
ok
non non
fumer
Musée des
Systèmes
Formels
Mausolée
Herbrand
Quercus
Böhmii
LC'91

Luca’s Dijkstra font

Trained first as a professional musician, then did a PhD in theoretical physics

A few of his contributions:

- Definitional (“metacircular”) interpreters
- Continuations
- Polymorphic lambda-calculus
- Forsythe, a language with intersection types
- Syntactic control of interference -> Separation logic
- Intrinsic semantics of subtyping

John Reynolds

1935 - 2013

Typing

We add just *one new rule* to the typing relation
– the so-called "Rule of Subsumption":

$$\frac{\Gamma \vdash e \in S \qquad\qquad S <: T}{\Gamma \vdash e \in T} \quad (\text{T-S}\textsc{ub})$$

Subtyping

$$\frac{}{B <: B} \quad (\text{S-Base})$$

$$\frac{T_1 <: S_1 \qquad T_2 <: S_2}{S_1 \to S_2 <: T_1 \to T_2} \quad (\text{S-Arrow})$$

$$\frac{S_1 <: T_1 \qquad \ldots \qquad S_n <: T_n}{\{l_1{:}S_1,\ \ldots\ , l_n{:}S_n,\ k_1{:}U_1,\ \ldots\ , k_m{:}U_m\} <: \{l_1{:}T_1,\ \ldots\ , l_n{:}T_n\}} \quad (\text{S-Rcd})$$

Now our term typechecks! :-)

$$
\dfrac{
  \dfrac{
    \dfrac{x\colon \mathsf{Person} \vdash x \in \mathsf{Person}}{x\colon \mathsf{Person} \vdash x.\mathsf{n} \in \mathsf{String}}\;\scriptstyle(\text{T-Var})
  }{\cdot \vdash \lambda x{:}\mathsf{Person}.\ x.\mathsf{n} \in \mathsf{Person} \to \mathsf{String}}\;\scriptstyle(\text{T-Abs})
  \qquad
  \dfrac{\vdots}{\mathsf{Person} \to \mathsf{String} <: \mathsf{Student} \to \mathsf{String}}
}{\cdot \vdash \lambda x{:}\mathsf{Person}.\ x.\mathsf{n} \in \mathsf{Student} \to \mathsf{String}}\;\scriptstyle(\text{T-Sub})
\qquad
\dfrac{\vdots}{\cdot \vdash \mathsf{alice} \in \mathsf{Student}}
$$

$$
\dfrac{\cdot \vdash \lambda x{:}\mathsf{Person}.\ x.\mathsf{n} \in \mathsf{Student} \to \mathsf{String} \qquad \cdot \vdash \mathsf{alice} \in \mathsf{Student}}{\cdot \vdash (\lambda x{:}\mathsf{Person}.\ x.\mathsf{n})\ \mathsf{alice} \in \mathsf{String}}\;\scriptstyle(\text{T-App})
$$

Professor at MIT. PhD (1970) with John McCarthy on chess endgames(!).

Some big contributions to PL:

- Data abstraction
  - CLU language
- Semantics of subtyping
  - Liskov substitution principle (with Jeanette Wing)

(Also major c… …ributed systems. …

Turing awa…

Who was the very first?

One of the first women to earn a PhD in Computer Science.

Barbara Liskov

1939 -

PhD in CS, 1965 (Wisconsin)

Missed being the very first CS PhD by a few hours!

Sister Mary Kenneth Keller

1913 - 1985

But we’re not quite done…

This derivation…

$$
\dfrac{
\dfrac{
\dfrac{\dfrac{x\text{: Person} \vdash x \in \text{Person}}{x\text{: Person} \vdash x.\text{n} \in \text{String}}\;(\text{T-Var})}{\cdot \vdash \lambda x\text{:Person. } x.\text{n} \in \text{Person} \to \text{String}}\;(\text{T-Abs})
\qquad
\dfrac{\vdots}{\text{Person} \to \text{String} <: \text{Student} \to \text{String}}
}{\cdot \vdash \lambda x\text{:Person. } x.\text{n} \in \text{Student} \to \text{String}}\;(\text{T-Sub})
\qquad
\dfrac{\vdots}{\cdot \vdash \text{alice} \in \text{Student}}
}{\cdot \vdash (\lambda x\text{:Person. } x.\text{n})\ \text{alice} \in \text{String}}\;(\text{T-App})
$$

…doesn’t match the LHS of the beta-reduction rule:

$$
\dfrac{
\dfrac{\dfrac{\Delta_1}{\Gamma,\, x{:}T \vdash e_1 \in U}}{\Gamma \vdash \lambda x{:}T.\ e_1 \in T \to U}\;(\text{T-Abs})
\qquad
\dfrac{\Delta_2}{\Gamma \vdash e_2 \in T}
}{\Gamma \vdash (\lambda x{:}T.\ e_1)\ e_2 \in U}\;(\text{T-App})
\longrightarrow
\dfrac{[\Delta_2/x]\Delta_1}{\Gamma \vdash [e_2/x]e_1 \in U}
$$

New reduction rule!

(Plus a similar rule for when T-Sub appears between T-Rcd and T-Proj.)

$$
\dfrac{
\dfrac{
\dfrac{\Delta_1}{\Gamma \vdash e_1 \in S_1 \to S_2}
\qquad
\dfrac{\dfrac{\Delta_2}{T_1 <: S_1} \qquad \dfrac{\Delta_3}{S_2 <: T_2}}{S_1 \to S_2 <: T_1 \to T_2}\;\text{(S-Arrow)}
}{\Gamma \vdash e_1 \in T_1 \to T_2}\;\text{(T-Sub)}
\qquad
\dfrac{\Delta_4}{\Gamma \vdash e_2 \in T_1}
}{\Gamma \vdash e_1\ e_2 \in T_2}\;\text{(T-App)}
$$

$$\longrightarrow$$

$$
\dfrac{
\dfrac{
\dfrac{\Delta_1}{\Gamma \vdash e_1 \in S_1 \to S_2}
\qquad
\dfrac{\dfrac{\Delta_4}{\Gamma \vdash e_2 \in T_1} \qquad \dfrac{\Delta_2}{T_1 <: S_1}}{\Gamma \vdash e_2 \in S_1}\;\text{(T-Sub)}
}{\Gamma \vdash e_1\ e_2 \in S_2}\;\text{(T-App)}
\qquad
\dfrac{\Delta_3}{S_2 <: T_2}
}{\Gamma \vdash e_1\ e_2 \in T_2}\;\text{(T-Sub)}
$$

So… which is better?

Both!!

The **extrinsic** approach is appropriate when types truly are 'after-the-fact descriptions' of underlying untyped behavior

- e.g., gradual type systems for untyped languages

The **intrinsic** approach is needed when types "matter for meaning"…

- coercions between numeric types, strings, etc
- Haskell typeclasses, etc.
- record calculi

Must we choose?

No!

E.g., Liquid Haskell

- Intrinsic core (Haskell)
- Extrinsic refinement types

What I hope you got out of this talk

- The distinction between intrinsic (Church-style) and extrinsic (Curry-style) typing
  - and why it matters
- How it extends to langauges with subtyping
- A sense of a few important people
- Fun?

Thank you!!

Any more questions, discussion, …?