SUBTYPING EXTRAVAGANZA II

Natural Operators

What does:

"Hello" + 1 + 2

do?

What's going wrong?

$$\begin{tikzcd}
1 \arrow[d, "i", swap] \arrow[d, "toString"] & 2 \arrow[d, "i"] & \arrow[r, "add", "+"'] & 3 \arrow[d, "i", swap] \arrow[d, "toString"] \\
\text{"1"} & \text{"2"} & \arrow[r, "append", "+"'] & \text{"12"} \neq \text{"3"}
\end{tikzcd}$$

← not equal is the problem!

What is +?

$$\begin{tikzcd}
\text{Types} \arrow[r, "op"] \arrow[d, "\text{subtyping}"'] \arrow[dr, phantom, "="] & \text{Type} \arrow[d, "\text{subtyping}"] \\
\text{types} \arrow[r, "op'"] & \text{Type}
\end{tikzcd}$$

Unambiguity Categorically

$$\begin{tikzcd}
\mathrm{Sub}|_{in(op)} \arrow[r, "⟦-⟧", "\text{coerce}"'] & \text{Set} \arrow[d, "op"] \\
\mathrm{Sub}|_{out(op)} \arrow[r, "⟦-⟧", "\text{coerce}"'] & \text{Set}
\end{tikzcd}$$

Natural Transformation

$$\begin{tikzcd}
⟦I_1⟧ \arrow[r, "⟦op⟧"] \arrow[d, "\text{coerce}_{I_1 I_2}"'] \arrow[dr, phantom, "= \text{ commutative diagram}"] & ⟦O_1⟧ \arrow[d, "\text{coerce}_{O_1 O_2},\ O_1 <: O_2"] \\
⟦I_2⟧ \arrow[r, "⟦op⟧"] & ⟦O_2⟧
\end{tikzcd}$$

$I_1 <: I_2$

A Natural Transformation α from F to $G : C \to D$

1. For every object A of C, a morphism
$$F(A) \xrightarrow{\alpha_A} G(A)$$
2. such that for every $A \xrightarrow{f} B$ of C, we have
$$\begin{tikzcd} F(A) \arrow[r, "\alpha_A"] \arrow[d, "F(f)"'] \arrow[dr, phantom, "="] & G(A) \arrow[d, "G(f)"] \\ F(B) \arrow[r, "\alpha_B"'] & G(B) \end{tikzcd}$$

Example: First we need the $List : Set \to Set$

1. Given a set A, $List(A)$ is the set of lists of A
2. Given a function $f : A \to B$, $map(f) : List(A) \to List(B)$

3 & 4. Preserves identity and composition

So we have a functor!

Another Functor: The Id Functor

Given a category C, define Id_C as

1. $Id_C(A) = A$
2. $Id_C(f) = f$

3 & 4 are obvious

A Nat Trans Example: singleton

$singleton : Id_{set} \Rightarrow List^{:Set \to Set}$ is defined as

1. $singleton_A : A \to List(A)$
$= \lambda a.\ [a]$
2. $f : A \to B \Rightarrow$
$$\begin{tikzcd} A \arrow[r, "singleton_A"] \arrow[d, "f"'] \arrow[dr, phantom, "="] & List(A) \arrow[d, "map(f)"] \\ B \arrow[r, "singleton_B"'] & List(B) \end{tikzcd}$$
✓

Moving on to Types

Static Behavioral Subtyping

An expression of type A
can be used anywhere
an expression of type B
can be used
whenever $A <: B$

Counterexample: C#

Foo : List<int> and IList<double>

⇓

Broken

Foo (broken) IList<int> (works)

↓ ↙

val foo = new Foo(); ←

return foo.get(5);

Principality

T is the principal type of expression e if it can be used wherever any other type of e can be used

Challenges

singletonArray<T>(T) : Array<T>

singletonArray("Hello") has type ?

Array<String>

and... Array<Object>

and... Array<T> where String <: T

infinite set of incomparable types!!!