Subtyping

Consider n-ary products $\tau_1 \times \ldots \times \tau_n$

$$\frac{\forall i,\ \Gamma \vdash e_i : \tau_i}{(e_1, \ldots, e_n) : \tau_1 \times \ldots \times \tau_n} \qquad \frac{\Gamma \vdash e : \tau_1 \times \ldots \times \tau_n}{\Gamma \vdash \pi_i\ e : \tau_i}$$

π_i ← ith projection

$fst \triangleq \lambda x : int \times int.\ \pi_1 x$

$fst3 \triangleq \lambda x : int \times int \times int.\ \pi_1 x$

...

$fst\ (1,2,3)$ not well-typed but safe.

$\tau <: \tau'$ "τ is a subtype of τ'"

"Anything that expects a τ' can accept a τ."

$$\frac{\Gamma \vdash e : \tau \qquad \tau <: \tau'}{\Gamma \vdash e : \tau'}\ (\text{SUB})$$

"Anything of τ can behave like τ'"

← "subsumption rule"

$$\frac{}{\tau_1 \times \ldots \times \tau_n \times \ldots \times \tau_{n+k} <: \tau_1 \times \ldots \times \tau_n}\ (\text{SUB-WIDTH})$$

"width subtyping"

$$\frac{\cdot \vdash fst : int \times int \to int \qquad \dfrac{\cdot \vdash (1,2,3) : int \times int \times int \qquad \overline{int \times int \times int <: int \times int}}{\cdot \vdash (1,2,3) : int \times int}}{\cdot \vdash fst\ (1,2,3) : int}$$

Can we type $\cdot \vdash fst3\,(1,2)$? No.

$int \times int \not<: int \times int \times int$

$fstfst \triangleq \lambda x.(int \times int) \times int.\ \pi_1(\pi_1\, x)$

$$\dfrac{\cdot \vdash fstfst : (int \times int) \times int \qquad \cdot \vdash ((1,2,3),4) : (int \times int) \times int}{\cdot \vdash fstfst\ ((1,2,3),4)}$$

$$\dfrac{\forall i.\ \tau_i <: \tau_i'}{\tau_1 \times \ldots \times \tau_n <: \tau_1' \times \ldots \times \tau_n'}\ (\text{SUB-DEPTH}) \quad \text{"Depth Subtyping"}$$

$$\dfrac{\cdot \vdash ((1,2,3),4) : (int \times int \times int) \times int \qquad \dfrac{\dfrac{}{int \times int \times int <: int \times int}\ (\text{Width}) \qquad \dfrac{}{int <: int}\ ?}{(int \times int \times int) \times int <: (int \times int) \times int}\ (\text{Depth})}{\cdot \vdash ((1,2,3),4) : (int \times int) \times int}\ (\text{SUB})$$

Subtyping should be reflexive (and transitive)

2 ways to do:

1. Set up rules carefully so it's true

$$\dfrac{}{unit <: unit} \qquad \dfrac{}{int <: int} \quad \ldots$$

- Easier to implement

2. Add explicit rules

$$\dfrac{}{\tau <: \tau}\ (\text{SUB-REFL}) \qquad \dfrac{\tau_1 <: \tau_2 \qquad \tau_2 <: \tau_3}{\tau_1 <: \tau_3}\ (\text{SUB-TRANS})$$

Easier in theory

$\text{doproj} \triangleq \lambda f.(\text{int} \times \text{int} \times \text{int}) \to \text{int}.\ \lambda x.\text{int} \times \text{int} \times \text{int}.\ f\,x$

$\text{doproj} : ((\text{int} \times \text{int} \times \text{int}) \to \text{int}) \to \text{int} \times \text{int} \times \text{int} \to \text{int}$

Is doproj fst3 (1, 2, 3, 4) w.t.?
Yes. $\text{int} \times \text{int} \times \text{int} \times \text{int} <: \text{int} \times \text{int} \times \text{int}$

Is doproj fst3 (1,2) w.t.? No.

Is doproj fst (1,2,3) w.t?
No. We haven't defined subtyping on functions.
But should it be?

$(\text{int} \times \text{int}) \to \text{int} \overset{?}{<:} (\text{int} \times \text{int} \times \text{int}) \to \text{int}$
Yes. All it can do is π_1, π_2. Can handle any subtype of $\text{int} \times \text{int} \times \text{int}$

contravariant "fewer requirements"   covariant "more promises"

$$\frac{\tau_1' <: \tau_1 \qquad \tau_2 <: \tau_2'}{\tau_1 \to \tau_2 <: \tau_1' \to \tau_2'} \ (\text{SUB-FUN})$$

Progress, Preservation basically the same

Canonical Forms (Old)

...

n. If $\bullet \vdash e : \tau_1 \times \ldots \tau_n$ and e val then $e = (v_1, \ldots, v_n)$

Canonical Forms (w/subtyping)

n. If $\bullet \vdash e : \tau_1 \times \tau_n$ and e val then $e = (v_1, \ldots, v_m)$ for $m \geq n$

Lemma: If $\tau <: \tau_1 \times \dots \times \tau_n$ then $\tau = \tau_1' \times \dots \times \tau_m'$ for $m \geq n$

By induction on the deriv of $\tau <: \tau_1 \times \dots \times \tau_n$

Pf of CF. By induction on the derivation of $\cdot \vdash e : \tau_1 \times \dots \times \tau_n$

$$\frac{\forall i.\ \Gamma \vdash e_i : \tau_i}{\Gamma \vdash (e_1, \dots, e_n) : \tau_1 \times \dots \times \tau_n}$$ By inversion, e_1 val, ..., e_n val ✓

$$\frac{\Gamma \vdash e : \tau' \qquad \tau' <: \tau}{\Gamma \vdash e : \tau}\ (\text{SUB}).$$ By lemma, $\tau' = \tau_1' \times \dots \times \tau_m'$ for $m \geq n$.
By induction, $e = (v_1, \dots, v_k)$ for $k \geq m \geq n$. □