What is a monotone dependently typed function?

Andreas Abel[1] Sandro Stucki[2]

[1]Department of Computer Science and Engineering
Chalmers and Gothenburg University, Sweden

[2]PhD from EPFL Lausanne, Switzerland

Winter Meeting 2018 of the
Division of Functional Programming
Salt & Sill, Tjörn, Sweden
9 January 2018

Sandro Stucki

- PhD from EPFL Lausanne in November 2017

  Higher-Order Subtyping with Type Intervals

  Partial formalization of Scala's type system (on paper and in Agda).
    - Bounded type operators.
    - Lower and upper bounds (hence *Type Intervals*).
    - Not yet: variances.
- Other research interests:
    - Category theory.
    - Graph rewriting.
    - Systems biology.
    - Domain specific languages (esp. probablistic and stochastic systems).
- Is looking for a PostDoc position at Chalmers!

  https://sstucki.github.io/

Upper and lower bounds

- From the Java 9 standard library:

```
interface Map<K,V> {
  V computeIfAbsent
     ( K key
     , Function<? super K, ? extends V> mappingFunction
     );
}
```

- Presentation in *System F with bounded quantification*:

$$\forall\ \texttt{K V (K' } \geq \texttt{ K) (V' } \leq \texttt{ V)} \rightarrow$$
$$\texttt{Map K V} \rightarrow \texttt{K} \rightarrow \texttt{Function K' V'} \rightarrow \texttt{V}$$

- With *type intervals* ($* = \bot..\top$):

$$\texttt{(K : *) (V : *) (K' : K..}\top\texttt{) (V' : }\bot\texttt{..V)} \rightarrow$$
$$\texttt{Map K V} \rightarrow \texttt{K} \rightarrow \texttt{Function K' V'} \rightarrow \texttt{V}$$

Variance

- Scala has *variance annotations*:

  $\texttt{Function<-K,+V>} \cong \texttt{K} \to \texttt{V}$

  $\texttt{K'} \geq \texttt{K},\ \texttt{V'} \leq \texttt{V} \ \vdash\ \texttt{Function K' V'} \leq \texttt{Function K V}$

- With variances, no bounds needed:

  $\forall\ \texttt{K V} \to \texttt{Map K V} \to \texttt{Function K V} \to \texttt{K} \to \texttt{V}$

Variance and bounds

- Maps need comparable keys:

  ImmutMap< -K $\leq$ Comparable, +V >

  ImmutMap : (-K : $\bot$..Comparable) (+V : *) $\rightarrow$ *

- Precise kinding of identity:

  Id : (A : *) $\rightarrow$ A..A

- Identity should be covariant (monotone):

  Id : (+A : *) $\rightarrow$ A..A

  A $\leq$ B $\vdash$ Id A $\leq$ Id B

- Incomparable kinds!

  Id A : A..A   B..A $\leq$ A..A $\leq$ A..B

  Id B : B..B   B..A $\leq$ B..B $\leq$ A..B

Bounded higher-order variant subtyping

- Judgement $\Gamma \vdash F \leq G : K$

```
K ≤ K' ≤ Comparable, V' ≤ V ⊢
  Map ≤ Map : (-K : ⊥..Comparable) (+V : *) → *
  Map K'     ≤ Map K    : (+V : *) → *
  Map K' V' ≤ Map K V : *
```

- What about dependencies?

```
A : *, B : A..⊤ ⊢
  Id    ≤ Id    : (+X : *) → X..X
  Id A ≤ Id B : ??
```

- We wish for:

```
A : *, B : A..⊤ ⊢
  Id    ≤ Id    : (+X : *) → X..X
  Id A ≤ Id B : A..B
```

Variance in Type Theory

- Lowering everything one level:

```
Kinds          → Types
Types          → Terms
Type operators → Functions
Subtyping      → Partial order
```

- Examples:

```
id    : (+ n : ℕ) → n..n
minus : (+ n : ℕ) → (- i : 0..n) → 0..n
```

A falling paradigm

- Type theory paradigm:

  Types and terms share a typing context.

  If $\Gamma \vdash t : T$ then $\Gamma \vdash T : \mathrm{Type}$.

- Cannot work for variance:

  `+n :` $\mathbb{N} \vdash$ `minus n : (- i : 0..n)` $\rightarrow$ `0..n`

  but not

  `+n :` $\mathbb{N} \vdash$ `(- i : 0..n)` $\rightarrow$ `0..n : Type`

- Term is covariant in `n`, but its type mixed-variant!

Paradigm has fallen already

- Linearity [McBride 2016: I got plenty of nutting]

  While I can drink my beer only once (resource, linear),
  I can contemplate it over and over (mental object, unrestricted).

  ```
  (λ x → x) : (1 x : ℕ) → x..x
  ```

- Erasure [Barras/Bernardo, Sheard/Mishram-Linger 2008]: Irrelevant in the term, but relevant in the type.

  ```
  (λ n x xs → x :: xs) :
    (0 n : ℕ) (x : A) (xs : Vec A n) → Vec A (1 + n)
  ```

Idea for dependent variance

On the type side,
distinguish positive and negative occurrences of a variable.

- Substitute a type-side variable by *two* values!

  $\vdash$ `id : (+ n :` $\mathbb{N}$`)` $\rightarrow$ `n..n`
  $\vdash$ `3` $\leq$ `5 :` $\mathbb{N}$
  $\vdash$ `id 3` $\leq$ `id 5 : (n..n)[n:=(3,5)]`
  $\vdash$ `id 3` $\leq$ `id 5 : n[n:=(5,3)] .. n[n:=(3,5)]`
  $\vdash$ `id 3` $\leq$ `id 5 : 3..5`

- Same when *going types*: If $+n : \mathbb{N} \vdash t : T$ (or $-n : \mathbb{N} \vdash t : T$) then

$$-n^- : \mathbb{N}, +n^+ : \mathbb{N} \vdash T[n := (n^-, n^+)] : \text{Type}$$

Related idea

- Decompose mixed variance into co- and contravariance.
- Forces monotonicity.
- Example: negative data types.

  `data D = Lam (D → D)`

  $D = \mu(X^-, X^+).\ (X^- \to X^+)$

Applications of dependent variance

- Subtyping dependent types.
- Sized types $\mathsf{Nat} : (+i : \mathsf{Size}) \to \mathsf{Type}$.
- Explaining Agda's positivity checker.

Future work

- Does it work?
- Work out the details.
- Complete Agda formalization.
- Write a paper.

Related work

- Andreas Nuyts et al.: Modal models of type theory: Parametricity, irrelevance.
- Models types as categories instead of groupoids.