Higher-Order Subtyping, Revisited
Syntactic Completeness Proofs for Algorithmic Judgements

Andreas Abel
TYPES Workshop, April 21, 2006

Contents
1. Subtyping for type constructors ($F^\omega$)

2. Proof Technique for Metatheory
   • Elementary (no model)
   • Works for weak theories: STL, LF

1 Higher-Order Subtyping

Subtyping for Collections

• When a Float is expected, an Int is acceptable.

\[ \text{Int} \leq \text{Float} \]

• Read-only collections: a list of Ints passes for a list of Floats.

\[ \frac{\text{Int} \leq \text{Float}}{\text{List Int} \leq \text{List Float}} \]

• Mutable collections: cannot store a Float into an Int cell.

\[ \frac{\text{not} \ \text{Int} \leq \text{Float}}{\text{Array Int} \not\leq \text{Array Float}} \]

Subtyping and Variance

• Distinguish type constructors by their variance

\[
\begin{array}{lll}
\text{Array} & : & * \rightarrow * \quad \text{mixed-variant} \\
\text{List} & : & * \rightarrow * \quad \text{covariant} \\
\text{Sink} & : & * \rightarrow * \quad \text{contravariant}
\end{array}
\]
Subtyping applications:

\[
F : * \rightarrow * \quad A = B \\
F A \leq F B
\]

\[
F : * \rightarrow * \quad A \leq B \\
F A \leq F B
\]

\[
F : * \rightarrow * \quad B \leq A \\
F A \leq F B
\]

Polarized \( F^\circ \)

- Polarities
  \[ p ::= \circ | + | - \]

- Kinds
  \[ \kappa ::= * | \kappa \rightarrow \kappa' \]

- Type constructors
  \[ F, G ::= C | X | \lambda X.F | F G \]

- Constants \( C \), e.g.,
  \[
  \times : * \rightarrow * \rightarrow * \\
  \rightarrow : * \rightarrow * \rightarrow * \\
  \forall_\kappa : (\kappa \rightarrow *) \rightarrow *
  \]

Polarized Kinding

- Polarized contexts
  \[ \Gamma ::= \emptyset | \Gamma, X : p\kappa \]

- Polarized kinding
  \[ \Gamma \vdash F : \kappa \]

- E.g.,
  \[
  F : \circ(* \rightarrow *), \\
  X : -*, \\
  Y : ** \vdash FX \rightarrow FY : *
  \]

Declarative Equality and Subtyping

- Judgements
  \[
  \Gamma \vdash F = F' : \kappa \quad \beta\eta\text{-equality} \\
  \Gamma \vdash F \leq F' : \kappa \quad \text{polarized subtyping}
  \]

- Subtyping axioms, e.g., \( \Gamma \vdash \text{Array} \leq \text{List} : * \rightarrow * \).

- Axioms for \( \beta \) and \( \eta \).
• Reflexivity, transitivity, (anti)symmetry.

• Closure under abstraction and *application*.

\[
\begin{align*}
\Gamma \vdash F : \kappa \to \kappa' & \quad & \Gamma \vdash F : \kappa \circ \kappa' \\
\Gamma \vdash G \leq G' : \kappa & \quad & \Gamma \vdash G = G' : \kappa \\
\Gamma \vdash FG \leq FG' : \kappa' & \quad & \Gamma \vdash FG = FG' : \kappa'
\end{align*}
\]

Algorithmic Subtyping

• Judgement for *algorithmic subtyping*

\[\Gamma \vdash F \leq F' \equiv \kappa\]

• Steps

\[
\begin{align*}
\text{Array} \leq (\lambda X. \text{List} X) & \quad \equiv \quad * \to * \quad \text{apply down to kind } *: \\
\text{Array} Y \leq (\lambda X. \text{List} X) \; Y & \quad \equiv \quad * \quad \text{weak head normalize:} \\
\text{Array} Y \leq \text{List} Y & \quad \equiv \quad * \quad \text{compare heads (axiom):} \\
\text{Array} \leq \text{List} : * \to * & \quad \equiv \quad * \\
Y \leq Y & \quad \equiv \quad *
\end{align*}
\]

Kind-directed Algorithmic Subtyping

• Weak head normal forms

\[
\begin{align*}
N & ::= C \mid X \mid NG & \text{neutral (atomic)} \\
W & ::= N \mid \lambda XF & \text{weak head normal}
\end{align*}
\]

• Weak head evaluation

\[F \downarrow W\]

• Kind-directed algorithmic subtyping

\[
\begin{align*}
\Gamma \vdash F \leq F' \equiv \kappa & \quad \text{checking mode} \\
\Gamma \vdash N \leq N' \Rightarrow \kappa & \quad \text{inference mode}
\end{align*}
\]

• (Analogously for algorithmic equality)

Rules for Algorithmic Subtyping

• Checking mode

\[
\begin{align*}
\Gamma, X : p\kappa \vdash F \; X \leq F' \; X \equiv \kappa' \\
\Gamma \vdash F \leq F' \equiv p\kappa \to \kappa'
\end{align*}
\]

\[
\begin{align*}
F \downarrow N & \quad F' \downarrow N' & \quad \Gamma \vdash N \leq N' \Rightarrow * \\
\Gamma \vdash F \leq F' \equiv *
\end{align*}
\]
Inference mode: Axioms +

\[
\frac{(X:p\kappa) \in \Gamma}{\Gamma \vdash X \leq X \equiv \kappa}
\]

\[
\frac{\Gamma \vdash N \leq N' \Rightarrow +\kappa \rightarrow \kappa'}{\Gamma \vdash G \leq G' \equiv \kappa'}
\]

Completeness of Algorithmic Subtyping

- Soundness of algorithmic judgements easy
- Transitivity, (anti)symmetry easy
- Completeness hard: *Closure under application?*

Alternatives:
1. From strong normalization (Aspinall Hofmann 2005; Goguen 2005)
2. Model (e.g., Harper Pfenning 2004)
3. *Direct, syntactically*

From a Bird’s Perspective

- Type language of $\mathbb{F}^\omega$ is weak (no recursion)
- Roughly simply-typed $\lambda$-calculus
- Proof theory says: there is an elementary meta theory
- *How to construct this elementary proof?*
- Technical skill required

Main Lemma: Application and Substitution

- Let $\Gamma \vdash G \leq G' \equiv \kappa$. Prove simultaneously:
  1. If $\Gamma \vdash F \leq F' \equiv +\kappa \rightarrow \kappa'$ then $\Gamma \vdash FG \leq FG' \equiv \kappa'$.
  2. If $\Gamma, X : +\kappa \vdash N \leq N' \equiv \kappa'$ then
     - either $\Gamma \vdash [G/X]N \leq [G/X]N' \Rightarrow \kappa'$,
     - or $\Gamma \vdash [G/X]N \leq [G/X]N' \equiv \kappa'$ and $|\kappa'| \leq |\kappa|$.
  3. If $\Gamma, X : +\kappa \vdash F \leq F' \equiv \kappa'$ then $\Gamma \vdash [G/X]F \leq [G'/X]F' \equiv \kappa'$.
- Lexicographic induction on $|\kappa|$ and derivation length.
1. ...

2. Case $N = N' = Y \neq X$. Case $N = N' = X$. Case

$$\Gamma \vdash M \leq M' \Rightarrow \kappa'' \rightarrow \kappa' \quad \Gamma \vdash H \leq H' \Rightarrow \kappa''$$

$$\Gamma \vdash \text{M} \text{H} \leq M' \text{H'} \Rightarrow \kappa'$$

**Consequences and Evaluation**

Consequences of Main Lemma:

- Closure under $\beta$ and application.
- Reflexivity.
- Completeness.

Evaluation of proof:

- Short, direct
- Purely syntactical
- Avoiding logical relations and models
- Well-suited for formalization (e.g., in Twelf)

**Applicability of Proof Technique**

- Normalization of simply-typed lambda-calculus (Joachimski Matthes 2003)
- Algorithmic equality for LF
- Other logical frameworks (LLF, CLF)
- Predicative polymorphism!
- Languages of low proof-theoretical complexity
- POPLmark challenges
- Limitations
  - Impredicativity
  - Inductive types
Related Work

- Cut elimination for FOL
- Troelstra 1973: Syntactical normalization proof
- Joachimski Matthes 2003: $\lambda$ + permutative conversions
- Hereditary substitutions:
  - Watkins Cervesato Pfenning Walker 2003: Concurrent LF
  - Nanevski Pfenning Pientka 2005: Contextual Modal Type Theory
  - Adams (PhD 2005): $\lambda$-free LF
- Goguen 1995-2005: Typed Operational Semantics

Conclusions

- Purely syntactical approach to meta theory
- Does not work for CC or inductive types
- But applicable to many logical frameworks
- Proofs suited for formalization (HOAS)
- Should be in your tool box!