Polarized Subtyping for Sized Types

Andreas Abel

Computer Science Symposium Russia
June 10, 2006

Outline

Contents

1 Applications of Subtyping

1.1 Type-Based Termination

Type-Based Termination

- Termination for higher-order functional programs.
- Recursive function definition:

$$
\begin{array}{lcl}
f : \forall i.\, \mathsf{List}^i\, A \to B & & \\
f\ (x : \mathsf{List}^{i+1}\, A) & = & \ldots (f : \mathsf{List}^i\, A \to B)(e) \ldots \\
 & & \ldots g(f : \mathsf{List}^i\, A \to B) \ldots
\end{array}
$$

- Sized type: $\mathsf{List}^i\, A$ contains lists of length $< i$.
- Subtyping: $\mathsf{List}^i\, A \leq \mathsf{List}^{i+1}\, A \leq \cdots \leq \mathsf{List}^\infty\, A$.

1.2 Subtyping Collections

Subtyping for Collections

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

$$\mathsf{Int} \leq \mathsf{Float}$$

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

$$\frac{\mathsf{Int} \leq \mathsf{Float}}{\mathsf{List\ Int} \leq \mathsf{List\ Float}}$$

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

$$\textit{not}\ \frac{\mathsf{Int} \leq \mathsf{Float}}{\mathsf{Array\ Int} \leq \mathsf{Array\ Float}}$$

Unsound subtyping in JAVA

This code passes type-checking.

```
void init(Object[] a) {
    a[0] = new Object();
}

void main() {
    Integer[] a = new Integer[1];
    init(a);
    print(a[0].intValue());
}
```

But *throws* `ArrayStoreException`!

Subtyping and Variance

- Distinguish type constructors by their *variance*

Array	:	$* \overset{\circ}{\to} *$	mixed-variant
List	:	$* \overset{+}{\to} *$	covariant
Sink	:	$* \overset{-}{\to} *$	contravariant

- Subtyping applications:

$$\frac{F : * \overset{\circ}{\to} * \qquad A = B}{F\,A \leq F\,B}$$

$$\frac{F : * \overset{+}{\to} * \qquad A \leq B}{F\,A \leq F\,B} \qquad \frac{F : * \overset{-}{\to} * \qquad B \leq A}{F\,A \leq F\,B}$$

2 Polarized Subtyping

2.1 Types as λ-Expressions

Types as λ-Expressions

- Creating type constructors by *abstraction*:

$$\lambda X.F$$

- *Application* of a type constructor

$$F\,G$$

- Representation of $\forall A.\ A \rightarrow \mathsf{List}\ A$

$$\forall\,(\lambda A.\,((\rightarrow)\,A\,(\mathsf{List}\ A)))$$

- Equations

$$\begin{array}{lll} (\beta) & (\lambda X.F)\,G = [G/X]F & \\ (\eta) & \lambda X.\,(F\,X) = F & \text{if } X \text{ does not appear in } F \end{array}$$

2.2 Kinds and Polarities

Polarized F^{ω}

- Type constructors

$$F, G ::= C \mid X \mid \lambda X.F \mid F\,G$$

- Kinds

$$\kappa ::= * \mid \kappa \xrightarrow{p} \kappa'$$

- Polarities

$$p ::= \circ \mid + \mid -$$

- Assign kinds, e.g., to constants C:

$$\begin{array}{lcl} \times & : & * \xrightarrow{+} * \xrightarrow{+} * \\ \rightarrow & : & * \xrightarrow{-} * \xrightarrow{+} * \\ \forall_{\kappa} & : & (\kappa \xrightarrow{\circ} *) \xrightarrow{+} * \end{array}$$

Polarized Kinding

- Polarized contexts

$$\Gamma ::= \diamond \mid \Gamma, X : p\kappa$$

- Polarized kinding

$$\Gamma \vdash F : \kappa$$

- E.g.,

$$\begin{array}{ll} F : \circ(* \xrightarrow{+} *), & \\ X : -*, & \\ Y : +* & \vdash \quad F\,X \to F\,Y : * \end{array}$$

2.3 Equality and Subtyping

Declarative Equality and Subtyping

- Judgements

$$\begin{array}{ll} \Gamma \vdash F = F' : \kappa & \beta\eta\text{-equality} \\ \Gamma \vdash F \leq F' : \kappa & \text{polarized subtyping} \end{array}$$

- Subtyping axioms, e.g., $\Gamma \vdash \mathsf{Array} \leq \mathsf{List} : * \xrightarrow{+} *$.
- Axioms for β and η.
- Reflexivity, transitivity, (anti)symmetry.
- Closure under abstraction and *application*.

$$\frac{\Gamma \vdash F : \kappa \xrightarrow{+} \kappa' \qquad \Gamma \vdash G \leq G' : \kappa}{\Gamma \vdash F\,G \leq F\,G' : \kappa'} \qquad \frac{\Gamma \vdash F : \kappa \xrightarrow{\circ} \kappa' \qquad \Gamma \vdash G = G' : \kappa}{\Gamma \vdash F\,G = F\,G' : \kappa'}$$

3 Algorithmic Subtyping

Algorithmic Subtyping

- Judgement for *algorithmic subtyping*

$$\Gamma \vdash F \leq F' \leftleftarrows \kappa$$

- Steps

$$\begin{array}{lllll} \mathsf{Array} & \leq (\lambda X.\, \mathsf{List}\,X) & \leftleftarrows & * \xrightarrow{+} * & \text{apply down to kind } *\text{:} \\ \mathsf{Array}\,Y & \leq (\lambda X.\, \mathsf{List}\,X)\,Y & \leftleftarrows & * & \text{weak head normalize:} \\ \mathsf{Array}\,Y & \leq \mathsf{List}\,Y & \leftleftarrows & * & \text{compare heads (axiom):} \\ \quad \mathsf{Array} & \leq \mathsf{List} : * \xrightarrow{+} * & & & \text{continue with arguments:} \\ \quad Y & \leq Y \leftleftarrows * & & & \end{array}$$

Kind-directed Algorithmic Subtyping

- Weak head normal forms

$$
\begin{array}{lcll}
N & ::= & C\,\vec{G} \mid X\,\vec{G} & \text{neutral (atomic)} \\
W & ::= & N \mid \lambda X.F & \text{weak head normal}
\end{array}
$$

- Weak head evaluation (β-steps)

$$F \searrow W$$

- Kind-directed algorithmic subtyping

$$
\begin{array}{ll}
\Gamma \vdash F \leq F' \leftleftarrows \kappa & \text{checking mode} \\
\Gamma \vdash N \leq N' \rightrightarrows \kappa & \text{inference mode}
\end{array}
$$

Rules for Algorithmic Subtyping

- Checking mode

$$\frac{\Gamma, X\!:\!p\kappa \vdash F\,X \leq F'\,X \leftleftarrows \kappa'}{\Gamma \vdash F \leq F' \leftleftarrows p\kappa \to \kappa'}$$

$$\frac{F \searrow N \qquad F' \searrow N' \qquad \Gamma \vdash N \leq N' \rightrightarrows *}{\Gamma \vdash F \leq F' \leftleftarrows *}$$

- Inference mode: Axioms +

$$\frac{(X\!:\!p\kappa) \in \Gamma \qquad p \in \{\circ, +\}}{\Gamma \vdash X \leq X \rightrightarrows \kappa}$$

$$\frac{\Gamma \vdash N \leq N' \rightrightarrows +\kappa \to \kappa' \qquad \Gamma \vdash G \leq G' \leftleftarrows \kappa}{\Gamma \vdash N\,G \leq N'\,G' \rightrightarrows \kappa'}$$

4 Completeness through Cut-Elimination

Completeness of Algorithmic Subtyping

Show: 2 kinds of *cuts* are admissible.

1. Transitivity (easy inductive proof)

$$\frac{\Gamma \vdash F_1 \leq F_2 \leftleftarrows \kappa \qquad \Gamma \vdash F_2 \leq F_3 \leftleftarrows \kappa}{\Gamma \vdash F_1 \leq F_3 \leftleftarrows \kappa}$$

2. *Application* (non-trivial)

$$\frac{\Gamma \vdash F \leq F' \leftleftarrows \kappa \stackrel{+}{\to} \kappa' \qquad \Gamma \vdash G \leq G' \leftleftarrows \kappa}{\Gamma \vdash F\,G \leq F'\,G' \leftleftarrows \kappa}$$

3. Proof alternatives:
   (a) From strong normalization (Aspinall Hofmann 2005; Goguen 2005)
   (b) Kripke-Model (e.g., Harper Pfenning 2004)
   (c) *Direct, syntactically*

Towards a Direct Proof of Application

- Propositional sequent calculus: structural proof of cut elimination.
- Transfer to implicational natural deduction:
  1. Kinds = implicational propositions.
  2. Application of constructors = modus ponens.
  3. Substitution = cut elimination.

Adaption of Cut Elimination

Application is a direct consequence of:

Lemma 1 (Substitution)**.** *If* $\Gamma, X : +\kappa \vdash F \leq F' \leftleftarrows \kappa'$ *and* $\Gamma \vdash G \leq G' \leftleftarrows \kappa$*, then* $\Gamma \vdash [G/X]F \leq [G'/X]F' \leftleftarrows \kappa'$.

Proof. By lexicographic induction on κ and size of the first derivation. □

5 Conclusion

Related Work

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

Conclusions

- Polarities arise naturally when subtyping constructors
- Idea of algorithm: apply down to base kind
- Direct and short completeness proof
- Next: extend to bounded quantification