Calculi with coercive subtyping

{libor,xkollar2}@fi.muni.cz,
{ipeterka,rysavy,kolar}@fit.vutbr.cz

Motivation

Current state:

- Subtyping was studied extensivelly for systems with dependent types, most notably by Aspinall, Luo, Chen.
- Coercions are implemented in proof systems (LEGO, Coq, Plastic).

Intended contribution:

- Find a substantial form of subtyping that can "live" in systems of lambda cube and does not harm the desired properties.
- Make metatheoretical examination of these systems easier (transitivity, coherence, dependence between rules).
- Allow for incremental development of calculi by extending the basic subtyping systems in a "safe way".
- Apply the method to design of a calculus with dependent types and subtyping.

Subtyping

Subtyping judgement

$$\Gamma \vdash A \leq B$$

"More intuitionistic" view: subtyping witnessed by coercion

$$\Gamma \vdash \kappa : A \leq B$$

Coercions

Simple coercions are just insertive mappings

$$\kappa_{en} : EvenNats \hookrightarrow Nats$$

Parametric coercions are lifted mappings

parameterized either by types (in $\lambda\underline{\omega}_{\leq}$)

$$\kappa_{bt} : BinTree \leq Tree$$
$$\kappa_{bt}\ \alpha : BinTree\ \alpha \hookrightarrow Tree\ \alpha$$

or by values (in $\lambda P_{\leq}$)

$$\kappa_{vb} : Vec \leq Bag$$
$$\kappa_{vb}\ n : Vec\ n \hookrightarrow Bag\ n$$

Approach

- Take coercive subtyping as a fundamental concept.
- Every new type comes with a subtyping rule.
- Subtyping rule of arrow type:

$$\frac{\Gamma \vdash \kappa_1 : A' \hookrightarrow A \qquad \Gamma \vdash \kappa_2 : B \hookrightarrow B'}{\Gamma \vdash (\lambda f{:}A{\to}B.\kappa_2 \circ f \circ \kappa_1) : (A \to B) \hookrightarrow (A' \to B')}\ \rightarrow\text{-SUB}$$

- What form should coercion terms have?
- We do not have general subsumption rule, rather subsumption is done when it is really necessary.

Coercion inference problem

- Coercion involvement (subsumption) is limited to certain rules only.
- Functional application is a suitable one:

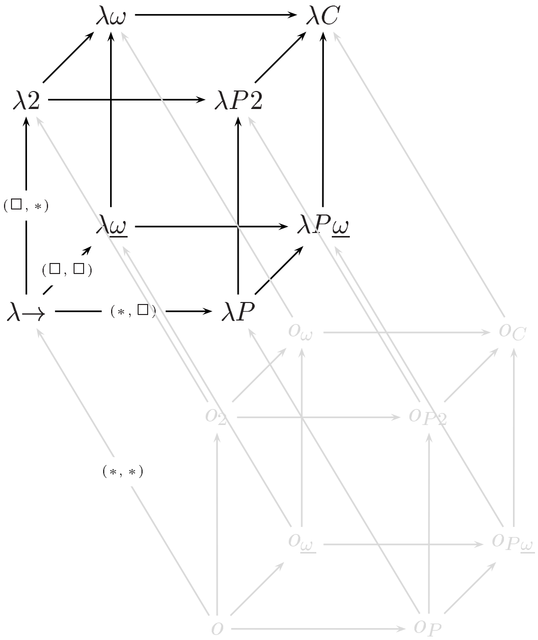
$$\frac{\Gamma \vdash M : \Pi x{:}A.B \qquad \Gamma \vdash \kappa : A' \leq A \qquad \Gamma \vdash N : A'}{\Gamma \vdash M\ N : [x := N]B}$$

Coercion inference algorithm:

- Input: A, A', Γ
- Output: κ

Use the output of the algorithm to make all coercions explicit. The resulting term is typeable in the type system without subtyping.

The context of λ-cube



Minimal System $o_{\leq}$

A λ-free fragment common to all λ-cube calculi with coercive subtyping.

$$\textsc{Γ-empty}\ \frac{}{\langle\rangle \vdash \star} \qquad \textsc{Γ-term}\ \frac{\Gamma \vdash A : \star}{\Gamma, x{:}A \vdash \star} \qquad \textsc{Γ-type}\ \frac{\Gamma \vdash \star}{\Gamma, \alpha{:}\star \vdash \star} \qquad \textsc{Γ-sub}\ \frac{\Gamma \vdash A : \star}{\Gamma, \kappa{:}\alpha{\leq}A \vdash \star}$$

$$\textsc{Γ-var-type}\ \frac{\Gamma \vdash \star \qquad \alpha{:}\star \in \Gamma}{\Gamma \vdash \alpha : \star} \qquad \textsc{Γ-var-term}\ \frac{\Gamma \vdash \star \qquad x{:}A \in \Gamma}{\Gamma \vdash x : A} \qquad \textsc{Γ-var-sub}\ \frac{\Gamma \vdash \star \qquad \kappa{:}\alpha{\leq}A \in \Gamma}{\Gamma \vdash \kappa : \alpha \leq A}$$

$$\textsc{s-tran}\ \frac{\Gamma \vdash \kappa_2 : \alpha_2 \leq A \qquad \kappa_1{:}\alpha_1{\leq}\alpha_2 \in \Gamma}{\Gamma \vdash \kappa_1 \circ \kappa_2 : \alpha_1 \leq A} \qquad \iota\textsc{-sub}\ \frac{\Gamma \vdash A : \star}{\Gamma \vdash \iota_A : A \leq A}$$

Calculi with subtyping

Π-FORM

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash \Pi x{:}A.B : s_2} \quad s_1, s_2 \in \{\star, \Box\}$$

Π-INTRO

$$\frac{\Gamma, x{:}A \vdash M : B \qquad \Gamma \vdash \Pi x{:}A.B : s}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B} \quad s \in \{\star, \Box\}$$

Π-ELIM

$$\frac{\Gamma \vdash M : \Pi x{:}A'.B \qquad \Gamma \vdash N : A \qquad \Gamma \vdash \kappa : A \leq A'}{\Gamma \vdash M\,N : [x := N]B}$$

Π-SUB

$$\frac{\Gamma \vdash \kappa_1 : A \leq A' \qquad \Gamma, x{:}A \vdash \kappa_2 : B \leq B'}{\Gamma \vdash \lambda(f : \Pi x{:}A.B).\kappa_2 \circ f \circ \kappa_1 : \Pi x{:}A'.B \leq \Pi x{:}A.B'}$$

λI-SUB

$$\frac{\Gamma \vdash A : s \qquad \Gamma, x{:}A \vdash K : \Box \qquad \Gamma, x{:}A \vdash B, B' : K \qquad \Gamma, x{:}A \vdash \kappa : B \leq B'}{\Gamma \vdash \lambda x{:}A.\kappa : (\lambda x{:}A.B) \leq (\lambda x{:}A.B')}$$

$$\text{where } s \in \begin{cases} \emptyset & \text{in } \lambda\!\rightarrow,\ \lambda 2 \\ \{\star\} & \text{in } \lambda P,\ \lambda P2 \\ \{\Box\} & \text{in } \lambda\underline{\omega},\ \lambda\omega \\ \{\star, \Box\} & \text{in } \lambda P\underline{\omega},\ \lambda C \end{cases}$$

λE-SUB

$$\frac{\Gamma \vdash C : \Pi x{:}A.K \qquad \Gamma \vdash C' : \Pi x{:}A.K \qquad \Gamma \vdash \kappa : C \leq C' \qquad \Gamma \vdash M : A}{\Gamma \vdash \kappa\, M : C\, M \leq C'\, M}$$

Example 1 ($\lambda\underline{\omega}_{\leq}$)

If every α-valued list can be viewed as an α-valued bag (multiset), then the type constructor List is a subtype of type constructor Bag.

$$\dfrac{\dfrac{\dfrac{\kappa : \mathit{List} \leq \mathit{Bag} \in \Gamma}{\Gamma, \alpha{:}\star \vdash \kappa : \mathit{List} \leq \mathit{Bag}} \qquad \Gamma, \alpha{:}\star \vdash \alpha : \star}{\Gamma, \alpha{:}\star \vdash \kappa\ \alpha : \mathit{List}\ \alpha \leq \mathit{Bag}\ \alpha}\ \lambda\text{E-SUB}}{\Gamma \vdash \lambda\alpha{:}\star.\kappa\ \alpha : \lambda\alpha{:}\star.\mathit{List}\ \alpha \leq \lambda\alpha{:}\star.\mathit{Bag}\ \alpha}\ \lambda\text{I-SUB}$$

Example 2 ($\lambda P_{\leq}$)

- Primitive coercions are introduced in the context in the form of $\kappa : \alpha \leq A : (\Pi x_1{:}A_1 \ldots x_n{:}A_n.\star)$.
- Coercion is a parametrized mapping: $\kappa : \pi x_1{:}A_1 \ldots x_n{:}A_n.\alpha\ x_1 \ldots x_n \to A\ x_1 \ldots x_n$.

If every vector of positive values can be viewed as a vector of the same length, then the type family of vectors of positive values is a subtype of type family of vectors of arbitrary values.

$$\dfrac{\dfrac{\dfrac{\kappa : PVec \leq Vec \in \Gamma}{\Gamma, n{:}nat \vdash \kappa : PVec \leq Vec} \qquad \Gamma, n{:}nat \vdash n : nat}{\Gamma, n{:}Nat \vdash \kappa\ n : PVec\ n \leq Vec\ n}\ \lambda\text{E-SUB}}{\Gamma \vdash \lambda n{:}Nat.\kappa\ n : \lambda n{:}Nat.PVec\ n \leq \lambda n{:}Nat.Vec\ n}\ \lambda\text{I-SUB}$$

Special Case:
Dependent-type calculus with simple coercions

Take $\lambda P_{\leq}$ and constrain subtyping to simple types.
We get a calculus called $\lambda P_{\hookrightarrow}$ with simple coercions.

Properties of this calculus:

- subject reduction
- strong normalization
- decidability of typechecking

Subtyping properties:

- transitivity elimination
- anti-symmetry
- coherence

Conclusion

- Development of a particular calculus can benefit from the regularity of its context.
- Careful choice of inference rules makes the calculus simpler.
- Substantional parts of proofs can be reused.

Future work

- More general introduction of primitive coercions (modelling multiple inheritance).
- Thorough inspection of all vertices of subtype-extended λ-cube.
- Step towards programming languages: including Σ-types, records, objects.

Coercion Terms for Subkinding

$$\lambda\underline{\omega}_{\leq} \quad (\Box, \Box) : \qquad \frac{\Gamma, \alpha{:}\star \vdash \kappa : K \leq K' \qquad \Gamma \vdash K, K' : \Box}{\Gamma \vdash \Lambda\alpha{:}\star.\kappa : (\Pi\alpha{:}\star.K) \leq (\Pi\alpha{:}\star.K')}$$

$$\lambda^{P}_{\leq} \quad (\star, \Box) : \qquad \frac{\Gamma \vdash \kappa_1 : A \leq A' \qquad \Gamma, x : A \vdash \kappa_2 : K \leq K'}{\Gamma \vdash \Lambda f{:}(\Pi x{:}A.K).\kappa_2 \circ f \circ \kappa_1 : \Pi x{:}A'.K \leq \Pi x{:}A.K'}$$

Coercive Subtyping

Simple Coercions – type $\leq$ type

-
- $even \leq nat$, $\pi x : nat.(A\ x) \leq \pi x : even.(A\ x)$

Parametrised Coercions – family of types $\leq$ family of types

-
- $\forall n{:}\star.List\ n \leq Bag\ n$

Dependent Coercions – type $\leq$ family of types

- Luo & Soloviev (1999)
- $l{:}List\ A \leq_c Vec\ A\ (len\ l)$