Mechanized undecidability of subtyping in System F

Roberto Álvarez
Advisor: Yannick Forster

Saarland University
Programming Systems Lab

Final Master's talk

09.05.2022

Bounded quantification

Combines type polymorphism with subtyping.

Terms and types of System $F_{\leqslant:}$:

$$s, t ::= x \mid \lambda_{x:\tau}.\, t \mid \Lambda_{\alpha \leqslant: \tau}.\, t \mid t\, s \mid t\, \tau$$

$$\sigma, \tau ::= \alpha \mid \sigma \to \tau \mid \forall_{\alpha \leqslant: \sigma}.\, \tau \mid \top$$

Bounded quantification

Combines type polymorphism with subtyping.

Terms and types of System $F_{\leqslant:}$:

$$s, t ::= x \mid \lambda_{x:\tau}.\, t \mid \Lambda_{\alpha \leqslant: \tau}.\, t \mid t\, s \mid t\, \tau$$

$$\sigma, \tau ::= \alpha \mid \sigma \to \tau \mid \forall_{\alpha \leqslant: \sigma}.\, \tau \mid \top$$

Bounded quantification

Combines type polymorphism with subtyping.

Terms and types of System $F_{\leqslant:}$:

$$s, t ::= x \mid \lambda_{x:\tau}.\, t \mid \Lambda_{\alpha \leqslant: \tau}.\, t \mid t\, s \mid t\, \tau$$

$$\sigma, \tau ::= \alpha \mid \sigma \to \tau \mid \forall_{\alpha \leqslant: \sigma}.\, \tau \mid \top$$

Unbounded quantification can be defined with $\top$:

$$\forall \alpha.\, \tau := \forall_{\alpha \leqslant: \top}.\, \tau$$

History

1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.

History

1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.

1990 Curien and Ghelli give a typechecking algorithm by means of a term rewritting system.
The algorithm is sound and complete by construction.

History

1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.

1990 Curien and Ghelli give a typechecking algorithm by means of a term rewritting system.
The algorithm is sound and complete by construction.

1990 Ghelli gives a proof of termination.
The proof turns out to be wrong.

History

1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.

1990 Curien and Ghelli give a typechecking algorithm by means of a term rewritting system.
The algorithm is sound and complete by construction.

1990 Ghelli gives a proof of termination.
The proof turns out to be wrong.

1992 Ghelli gives a counterexample.

History

1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.

1990 Curien and Ghelli give a typechecking algorithm by means of a term rewritting system.
The algorithm is sound and complete by construction.

1990 Ghelli gives a proof of termination.
The proof turns out to be wrong.

1992 Ghelli gives a counterexample.

1994 Pierce gives a proof of undecidability.

History

- 1985 System $F_{\leqslant:}$ is first introduced by Cardelli and Wegner.
- 1990 Curien and Ghelli give a typechecking algorithm by means of a term rewritting system.
  The algorithm is sound and complete by construction.
- 1990 Ghelli gives a proof of termination.
  The proof turns out to be wrong.
- 1992 Ghelli gives a counterexample.
- 1994 Pierce gives a proof of undecidability.
- 2022 Pierces's proof is mechanized.

Subtyping bounded quantifiers

$$\frac{\Gamma \vdash \tau_1 \leqslant: \sigma_1 \qquad \Gamma, \alpha \leqslant: \tau_1 \vdash \sigma_2 \leqslant: \tau_2}{\Gamma \vdash \forall_{\alpha \leqslant: \sigma_1}.\, \sigma_2 \leqslant: \forall_{\alpha \leqslant: \tau_1}.\, \tau_2} \textsf{All}$$

Subtyping bounded quantifiers

$$\frac{\Gamma \vdash \tau_1 \leqslant: \sigma_1 \qquad \Gamma, \alpha \leqslant: \tau_1 \vdash \sigma_2 \leqslant: \tau_2}{\Gamma \vdash \forall_{\alpha \leqslant: \sigma_1}. \sigma_2 \leqslant: \forall_{\alpha \leqslant: \tau_1}. \tau_2} \text{All}$$

we say that σ_2 gets *rebounded*.

$F_{\leqslant:}$ subtyping

$$\frac{\Gamma \vdash \tau_1 \leqslant: \sigma_1 \qquad \Gamma, \alpha \leqslant: \tau_1 \vdash \sigma_2 \leqslant: \tau_2}{\Gamma \vdash \forall_{\alpha \leqslant: \sigma_1}.\, \sigma_2 \leqslant: \forall_{\alpha \leqslant: \tau_1}.\, \tau_2} \text{ All}$$

$$\frac{\Gamma \vdash \tau_1 \leqslant: \sigma_1 \qquad \Gamma \vdash \sigma_2 \leqslant: \tau_2}{\Gamma \vdash \sigma_1 \to \sigma_2 \leqslant: \tau_1 \to \tau_2} \text{ Arrow}$$

$$\frac{}{\Gamma \vdash \tau \leqslant: \tau} \text{ Refl} \qquad \frac{}{\Gamma \vdash \tau \leqslant: \top} \text{ Top}$$

$$\frac{\Gamma \vdash \sigma \leqslant: \phi \qquad \Gamma \vdash \phi \leqslant: \tau}{\Gamma \vdash \sigma \leqslant: \tau} \text{ Trans} \qquad \frac{}{\Gamma \vdash \alpha \leqslant: \Gamma(\alpha)} \text{ Var}$$

$F_{\leqslant:}$ subtyping:
Given arbitrary Γ, σ and τ, is there a derivation of $\Gamma \vdash \sigma \leqslant: \tau$?

$F_{\leqslant:}$ typechecking

$$\frac{}{\Delta;\Gamma \vdash x : \Delta(x)}\ \text{Var} \qquad \frac{\Delta;\Gamma \vdash t : \sigma \qquad \Gamma \vdash \sigma \leqslant: \tau}{\Delta;\Gamma \vdash t : \tau}\ \text{Subsumption}$$

$$\frac{\Delta, x : \sigma;\Gamma \vdash t : \tau}{\Delta;\Gamma \vdash \lambda_{x:\sigma}.t\ : \sigma \to \tau}\ \text{Term-Abst} \qquad \frac{\Delta;\Gamma \vdash t : \sigma \to \tau \qquad \Delta;\Gamma \vdash u : \sigma}{\Delta;\Gamma \vdash t\ u : \tau}\ \text{Term-Inst}$$

$$\frac{\Delta;\Gamma, \alpha \leqslant: \sigma \vdash t : \tau}{\Delta;\Gamma \vdash \Lambda_{\alpha \leqslant: \sigma}.t\ : \forall_{\alpha \leqslant: \sigma}.\tau}\ \text{Type-Abst} \qquad \frac{\Delta;\Gamma \vdash t : \forall_{\alpha \leqslant: \sigma}.\tau \qquad \Gamma \vdash \sigma_1 \leqslant: \sigma}{\Delta;\Gamma \vdash t\ \sigma_1 : \tau[\sigma_1/\alpha]}\ \text{Type-Inst}$$

$F_{\leqslant:}$ typechecking:
Given arbitrary Δ,Γ, t and τ, is there a derivation of $\Delta;\Gamma \vdash t : \tau$?

Undecidability

Theorem

$F_{\leqslant:}$ subtyping is undecidable.

Theorem

$F_{\leqslant:}$ typechecking is undecidable.

Undecidability

Theorem

$F_{\leqslant:}$ subtyping is undecidable.

Proof.

By a chain of many-one reductions, Pierce [1994]:
2CM halting $\preceq_m$ RM halting $\preceq_m \cdots \preceq_m F_{\leqslant:}$ subtyping □

Theorem

$F_{\leqslant:}$ typechecking is undecidable.

Undecidability

Theorem

$F_{\leqslant:}$ subtyping is undecidable.

Proof.

By a chain of many-one reductions, Pierce [1994]:
2CM halting $\preceq_m$ RM halting $\preceq_m \cdots \preceq_m F_{\leqslant:}$ subtyping □

Theorem

$F_{\leqslant:}$ typechecking is undecidable.

Proof.

By reduction from subtyping; we give a term that is well-typed iff a subtyping statement holds:

$$\Gamma \vdash \sigma \leqslant: \tau \iff \Gamma \vdash (\Lambda_{\alpha \leqslant: \tau}.\lambda_{x:\alpha}.x)\ \sigma\ :\ \sigma \to \sigma$$

□

Undecidability

Theorem

$F_{\leqslant:}$ subtyping is undecidable.

Proof.

By a chain of many-one reductions, Pierce [1994]:
2CM halting $\preceq_m$ RM halting $\preceq_m \cdots \preceq_m F_{\leqslant:}$ subtyping □

Theorem

$F_{\leqslant:}$ typechecking is undecidable.

Proof.

By reduction from subtyping; we give a term that is well-typed iff a subtyping statement holds:

$$\Gamma \vdash \sigma \leqslant: \tau \iff \Gamma \vdash (\Lambda_{\alpha \leqslant: \tau}.\lambda_{x:\alpha}.x)\,\sigma \;:\; \sigma \to \sigma$$

□

Note: arrow types are only required on the second proof.

Overview

To show RM halting $\preceq_m F_{\leqslant:}$ subtyping Pierce shows:

$$R \text{ halts } \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

Overview

To show RM halting $\preceq_m F_{\leqslant:}$ subtyping Pierce shows:

$$R \text{ halts } \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Rightarrow$) By induction on the trace, in order to encode the stepping of the machine we need:

Overview

To show RM halting $\preceq_m F_{\leqslant:}$ subtyping Pierce shows:

$$R \text{ halts } \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Rightarrow$) By induction on the trace, in order to encode the stepping of the machine we need:

- To rebound the right hand side with an operator that *flips* inequalities using contravariance:

$$\overline{\tau} := \forall_{\alpha \leqslant: \tau}.\alpha$$

$$\Gamma \vdash \overline{\sigma} \leqslant: \overline{\tau} \iff \Gamma \vdash \tau \leqslant: \sigma \tag{1}$$

Overview

To show RM halting $\preceq_m F_{\leqslant:}$ subtyping Pierce shows:

$$R \text{ halts} \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Rightarrow$) By induction on the trace, in order to encode the stepping of the machine we need:

- To rebound the right hand side with an operator that *flips* inequalities using contravariance:

$$\overline{\tau} := \forall_{\alpha \leqslant: \tau}.\alpha$$

$$\Gamma \vdash \overline{\sigma} \leqslant: \overline{\tau} \iff \Gamma \vdash \tau \leqslant: \sigma \tag{1}$$

- To substitute variables eagerly, as the machine does:

$$\alpha \leqslant: \phi \vdash \sigma \leqslant: \tau \iff \vdash \sigma[\phi/\alpha] \leqslant: \tau[\phi/\alpha] \tag{2}$$

Does not hold in general, e.g. with $\phi = \sigma = \top$ and $\tau = \alpha$.

Overview

To show RM halting $\preceq_m F_{\leqslant:}$ subtyping Pierce shows:

$$R \text{ halts } \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Rightarrow$) By induction on the trace, in order to encode the stepping of the machine we need:

- Flip property:

$$\Gamma \vdash \overline{\sigma} \leqslant: \overline{\tau} \iff \Gamma \vdash \tau \leqslant: \sigma \tag{1}$$

- Eager substitution:

$$\alpha \leqslant: \phi \vdash \sigma \leqslant: \tau \iff \vdash \sigma[\phi/\alpha] \leqslant: \tau[\phi/\alpha] \tag{2}$$

Overview

$$R \text{ halts} \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Leftarrow$) We need to analyze the derivation, however:

- Transitivity is too general; the intermediate type is arbitrary, there might be infinitely many derivations.

Overview

$$R \text{ halts } \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Leftarrow$) We need to analyze the derivation, however:

- Transitivity is too general; the intermediate type is arbitrary, there might be infinitely many derivations.
- We need to obtain derivations deterministically, to match the behaviour of the machine.

Overview

$$R \text{ halts} \iff \vdash \sigma \leqslant: \mathcal{T}(R)$$

($\Leftarrow$) We need to analyze the derivation, however:

- Transitivity is too general; the intermediate type is arbitrary, there might be infinitely many derivations.
- We need to obtain derivations deterministically, to match the behaviour of the machine.
- The types are too general; we need an invariant on the syntax. We only care about types of the form of translated machines.

Overview

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

Pierce defines the intermediate systems to address the requirements:

Overview

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

Pierce defines the intermediate systems to address the requirements:

$F^N_{\leqslant:}$ Restricted transitivity and flip property.

Overview

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

Pierce defines the intermediate systems to address the requirements:

- $F^N_{\leqslant:}$ Restricted transitivity and flip property.
- $F^D_{\leqslant:}$ Deterministic subtyping and syntactic invariants.

Overview

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

Pierce defines the intermediate systems to address the requirements:

$F^N_{\leqslant:}$ Restricted transitivity and flip property.

$F^D_{\leqslant:}$ Deterministic subtyping and syntactic invariants.

$F^F_{\leqslant:}$ Eager substitution.

Overview

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

Pierce defines the intermediate systems to address the requirements:

- $F^N_{\leqslant:}$ Restricted transitivity and flip property.
- $F^D_{\leqslant:}$ Deterministic subtyping and syntactic invariants.
- $F^F_{\leqslant:}$ Eager substitution.

The systems are implemented with deBrujin indices, however are presented with named variables.

System $F^N_{\leqslant:}$ (normal)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m \boxed{F^N_{\leqslant:} \preceq_m F_{\leqslant:}}$$

Makes subtyping syntax directed:

$$\frac{}{\Gamma \vdash^0_N \alpha \leqslant: \alpha}\ \mathsf{NRefl} \qquad \frac{\Gamma \vdash^i_N \Gamma(\alpha) \leqslant: \tau}{\Gamma \vdash^{\mathsf{S}i}_N \alpha \leqslant: \tau}\ \mathsf{NVar}$$

System $F^N_{\leqslant:}$ (normal)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m \boxed{F^N_{\leqslant:} \preceq_m F_{\leqslant:}}$$

Makes subtyping syntax directed:

$$\frac{}{\Gamma \vdash^0_N \alpha \leqslant: \alpha}\ \textsf{NRefl} \qquad \frac{\Gamma \vdash^i_N \Gamma(\alpha) \leqslant: \tau}{\Gamma \vdash^{\mathsf{S}i}_N \alpha \leqslant: \tau}\ \textsf{NVar}$$

Theorem 1

$(\exists i.\, \Gamma \vdash^i_N \sigma \leqslant: \tau) \iff \Gamma \vdash \sigma \leqslant: \tau$

System $F^N_{\leqslant:}$ (normal)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m \boxed{F^N_{\leqslant:} \preceq_m F_{\leqslant:}}$$

Makes subtyping syntax directed:

$$\frac{}{\Gamma \vdash^0_N \alpha \leqslant: \alpha}\ \textsf{NRefl} \qquad \frac{\Gamma \vdash^i_N \Gamma(\alpha) \leqslant: \tau}{\Gamma \vdash^{\mathsf{S}i}_N \alpha \leqslant: \tau}\ \textsf{NVar}$$

Theorem 1

$$(\exists i.\, \Gamma \vdash^i_N \sigma \leqslant: \tau) \iff \Gamma \vdash \sigma \leqslant: \tau$$

The flip property is now immediate.

Lemma 2

$$\Gamma \vdash^{\mathsf{S}i}_N \overline{\sigma} \leqslant: \overline{\tau} \iff \Gamma \vdash^i_N \tau \leqslant: \sigma$$

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

The *w-fold polarized* syntax classifies positive and negative types:

$$\begin{aligned} \tau^+ &::= \top \mid \forall_{\alpha_0 \leqslant: \tau_0^-, \ldots, \alpha_w \leqslant: \tau_w^-} . \overline{\tau^-} \\ \tau^- &::= \alpha \mid \forall_{\alpha_0, \ldots, \alpha_w} . \overline{\tau^+} \end{aligned}$$

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

The *w-fold polarized* syntax classifies positive and negative types:

$$\begin{aligned} \tau^+ &::= \top \mid \forall_{\alpha_0 \leqslant: \tau_0^-, \ldots, \alpha_w \leqslant: \tau_w^-} . \overline{\tau^-} \\ \tau^- &::= \alpha \mid \forall_{\alpha_0, \ldots, \alpha_w} . \overline{\tau^+} \end{aligned}$$

The *w-fold polyadic* binders are the syntactic invariant required: machines have a constant number of registers that are updated simultaneously.

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

The *w-fold polarized* syntax classifies positive and negative types:

$$\begin{aligned}\tau^+ &::= \top \mid \forall_{\alpha_0 \leqslant:\tau_0^-,\ldots,\alpha_w \leqslant:\tau_w^-}.\ \overline{\tau^-} \\ \tau^- &::= \alpha \mid \forall_{\alpha_0,\ldots,\alpha_w}.\ \overline{\tau^+}\end{aligned}$$

The *w-fold polyadic* binders are the syntactic invariant required: machines have a constant number of registers that are updated simultaneously.

New quantifier rule:

$$\frac{\Gamma, \alpha_0 \leqslant:\phi_0^-, \ldots \alpha_w \leqslant:\phi_w^- \vdash^i_D \tau^- \leqslant:\sigma^+}{\Gamma \vdash^{\mathsf{S}i}_D \forall_{\alpha_0,\ldots,\alpha_w}.\ \overline{\sigma^+} \leqslant: \forall_{\alpha_0 \leqslant:\phi_0^-,\ldots,\alpha_w \leqslant:\phi_w^-}.\ \overline{\tau^-}}\ \textsf{DAllFlip}$$

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

We need a translation $[\![-]\!]$ from *well-scoped w-fold polyadic* syntax to *unscoped* syntax:

$$[\![\mathsf{var}_D\ i\ j]\!] = \mathsf{var}_N\ (\widehat{i} + w * \widehat{j})$$

where $i : \mathbb{I}^w$ and $j : \mathbb{I}^n$ for some n.

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

We need a translation $[\![-]\!]$ from *well-scoped w-fold polyadic* syntax to *unscoped* syntax:

$$[\![\mathsf{var}_D\ i\ j]\!] = \mathsf{var}_N\ (\widehat{i} + w * \widehat{j})$$

where $i : \mathbb{I}^w$ and $j : \mathbb{I}^n$ for some n.

Lemma 3

For all τ and polyadic substitution θ:

$$[\![\tau[\theta]]\!] = [\![\tau]\!][[\![\theta]\!]]$$

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

We need a translation $[\![-]\!]$ from *well-scoped w-fold polyadic* syntax to *unscoped* syntax:

$$[\![\mathsf{var}_D\ i\ j]\!] = \mathsf{var}_N\ (\widehat{i} + w * \widehat{j})$$

where $i : \mathbb{I}^w$ and $j : \mathbb{I}^n$ for some n.

Lemma 3

For all τ and polyadic substitution θ:

$$[\![\tau[\theta]]\!] = [\![\tau]\!][[\![\theta]\!]]$$

Proof.

By extensionality up to a bound. □

System $F^D_{\leqslant:}$ (deterministic)

$$\mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

Theorem

$(\exists i.\, \Gamma \vdash^i_D \sigma {\leqslant:} \tau) \iff (\exists j.\, [\![\Gamma]\!] \vdash^j_N [\![\sigma]\!] {\leqslant:} [\![\tau]\!])$

Proof.

($\Rightarrow$) By induction on the derivation.

System $F^D_{\leqslant:}$ (deterministic)

$$\text{RM} \preceq_m F^F_{\leqslant:} \preceq_m \boxed{F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}} \preceq_m F_{\leqslant:}$$

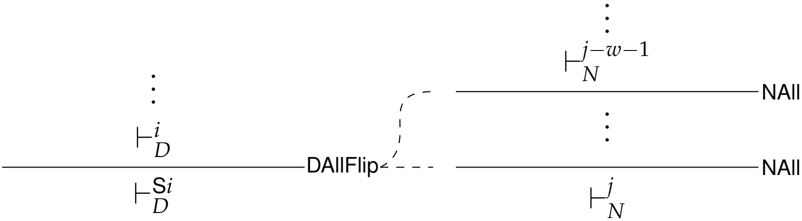
Theorem

$$(\exists i.\ \Gamma \vdash^i_D \sigma {\leqslant:} \tau) \iff (\exists j.\ [\![\Gamma]\!] \vdash^j_N [\![\sigma]\!] {\leqslant:} [\![\tau]\!])$$

Proof.

($\Rightarrow$) By induction on the derivation.

($\Leftarrow$) The new quantifier rule corresponds to $w + 1$ uses of the old rule, therefore we use complete induction on the height of the derivation.



System $F^D_{\leqslant:}$ (deterministic)

We can already show a generalization of eager substitution:

Lemma 4
For all i there is a j such that $j \leq i$ and:

$$\alpha_0 {\leqslant:} \phi_0, \ldots, \alpha_w {\leqslant:} \phi_w, \Gamma \vdash^i_D \sigma {\leqslant:} \tau$$

$$\Longleftrightarrow$$

$$\Gamma[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w] \vdash^j_D \sigma[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w] {\leqslant:} \tau[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w]$$

System $F^D_{\leqslant:}$ (deterministic)

We can already show a generalization of eager substitution:

Lemma 4
For all i there is a j such that $j \leq i$ and:

$$\alpha_0 \leqslant: \phi_0, \ldots, \alpha_w \leqslant: \phi_w, \Gamma \vdash^i_D \sigma \leqslant: \tau$$

$$\iff$$

$$\Gamma[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w] \vdash^j_D \sigma[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w] \leqslant: \tau[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w]$$

Proof.
Both directions follow by induction.
The proof involves substituting the closed types that were first introduced in a context, this motivates the use of well-scoped syntax. □

System $F^F_{\leqslant:}$ (flattened)

$$\mathsf{RM} \preceq_m \boxed{F^F_{\leqslant:} \preceq_m F^D_{\leqslant:}} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

The final variant incorporates eager substitution in the quantifier rule:

$$\frac{\vdash^i_F \tau[\phi_0/\alpha_0,\ldots,\phi_w/\alpha_w] \leqslant: \sigma[\phi_0/\alpha_0,\ldots,\phi_w/\alpha_w]}{\vdash^{\mathsf{S}i}_F \forall_{\alpha_0\leqslant:\top,\ldots,\alpha_w\leqslant:\top}.\ \overline{\sigma} \leqslant: \forall_{\alpha_0\leqslant:\phi_0,\ldots,\alpha_w\leqslant:\phi_w}.\ \overline{\tau}}\ \mathsf{FAllFlip}$$

Theorem 5

$(\exists i.\ \vdash^i_F \sigma \leqslant: \tau) \iff (\exists j.\ \vdash^j_D \sigma \leqslant: \tau)$

System $F^F_{\leqslant:}$ (flattened)

$$\text{RM} \preceq_m \boxed{F^F_{\leqslant:} \preceq_m F^D_{\leqslant:}} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

The final variant incorporates eager substitution in the quantifier rule:

$$\frac{\vdash^i_F \tau[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w] \leqslant: \sigma[\phi_0/\alpha_0, \ldots, \phi_w/\alpha_w]}{\vdash^{\mathsf{S}i}_F \forall_{\alpha_0 \leqslant: \top, \ldots, \alpha_w \leqslant: \top} .\ \overline{\sigma} \leqslant: \forall_{\alpha_0 \leqslant: \phi_0, \ldots, \alpha_w \leqslant: \phi_w} .\ \overline{\tau}} \ \text{FAllFlip}$$

Theorem 5

$(\exists i.\ \vdash^i_F \sigma \leqslant: \tau) \iff (\exists j.\ \vdash^j_D \sigma \leqslant: \tau)$

Proof.

($\Rightarrow$) By induction on the derivation.
($\Leftarrow$) The new quantifier rule skips all the instances of the variable rule, we use complete induction on the height of the derivation again. □

System $F^F_{\leqslant:}$ (flattened)

$$\boxed{\mathsf{RM} \preceq_m F^F_{\leqslant:}} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

We can show the reduction from RM halting.

Theorem 6

R halts $\iff \exists i.\ \vdash^i_F \sigma \leqslant: \mathcal{T}(R)$

System $F^F_{\leqslant:}$ (flattened)

$$\boxed{\mathsf{RM} \preceq_m F^F_{\leqslant:}} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

We can show the reduction from RM halting.

Theorem 6

R halts $\iff \exists i.\ \vdash^i_F \sigma \leqslant: \mathcal{T}(R)$

Proof.

(⇒) By induction on the trace.
(⇐) One step of the machine corresponds to two applications of the quantifier rule, once again we do complete induction on the height of the derivation. □

Typechecking

To show that subtyping reduces to typechecking it is enough to show:

Lemma 7

For all Γ, σ *and* τ*:*

$$\Gamma \vdash \sigma \leqslant: \tau \iff \Gamma \vdash (\Lambda_{\alpha \leqslant: \tau}.\lambda_{x:\alpha}.x)\,\sigma \;:\; \sigma \to \sigma$$

Proof.

($\Rightarrow$) By type instantiation.

Typechecking

To show that subtyping reduces to typechecking it is enough to show:

Lemma 7

For all Γ, σ *and* τ*:*

$$\Gamma \vdash \sigma \leqslant: \tau \iff \Gamma \vdash (\Lambda_{\alpha \leqslant: \tau}.\lambda_{x:\alpha}.x)\, \sigma \;:\; \sigma \to \sigma$$

Proof.

($\Rightarrow$) By type instantiation.
($\Leftarrow$) By inversion on the rules with induction on the height of the derivation in the subsumption case. $\square$

Summary

$$\mathsf{2CM} \preceq_m \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:} \text{ subtyping } \preceq_m F_{\leqslant:} \text{ typechecking}$$

Summary

$$\mathsf{2CM} \preceq_m \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:} \text{ subtyping } \preceq_m F_{\leqslant:} \text{ typechecking}$$

- Syntax directed subtyping is better suited to analyze derivations.

Summary

$$\mathsf{2CM} \preceq_m \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:} \text{ subtyping } \preceq_m F_{\leqslant:} \text{ typechecking}$$

- Syntax directed subtyping is better suited to analyze derivations.
- Polarized syntax enables eager substitution.

Summary

$$\mathsf{2CM} \preceq_m \ \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:} \text{ subtyping } \preceq_m F_{\leqslant:} \text{ typechecking}$$

- Syntax directed subtyping is better suited to analyze derivations.
- Polarized syntax enables eager substitution.
- Well-scoped polyadic syntax profiting from Autosubst2 features.

Summary

$$\mathsf{2CM} \preceq_m \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:}\ \text{subtyping} \preceq_m F_{\leqslant:}\ \text{typechecking}$$

- Syntax directed subtyping is better suited to analyze derivations.
- Polarized syntax enables eager substitution.
- Well-scoped polyadic syntax profiting from Autosubst2 features.
- Induction on height of derivations is required in most proofs.

Summary

$$\mathsf{2CM} \preceq_m \mathsf{RM} \preceq_m F^F_{\leqslant:} \preceq_m F^D_{\leqslant:} \preceq_m F^N_{\leqslant:} \preceq_m F_{\leqslant:}$$

$$F_{\leqslant:} \text{ subtyping } \preceq_m F_{\leqslant:} \text{ typechecking}$$

- Syntax directed subtyping is better suited to analyze derivations.
- Polarized syntax enables eager substitution.
- Well-scoped polyadic syntax profiting from Autosubst2 features.
- Induction on height of derivations is required in most proofs.
- Construction of subtyping judgements corresponds to a deterministic state transformation.

Summary of mechanization

	LOC	
	Spec.	Proof
Shared facts	500	400
Autosubst2 syntax:		
unscoped	130	20
well-scoped	200	150
Reductions:		
Subtyping $\preceq_m$ Typechecking	30	160
$F^N_{\leqslant:} \preceq_m F_{\leqslant:}$	30	60
$F^D_{\leqslant:} \preceq_m F^N_{\leqslant:}$	150	250
$F^F_{\leqslant:} \preceq_m F^D_{\leqslant:}$	50	100
$\mathsf{RM} \preceq_m F^F_{\leqslant:}$	80	120
$\mathsf{CM2} \preceq_m \mathsf{RM}$	50	120
Total	2580	

Future work

- Wehr and Thiemann [2009] reduce $F^{D}_{\leqslant:}$ subtyping to subtyping existential types with upper ($\exists x {\leqslant:} \tau.\ \sigma$) and lower ($\exists \tau {\leqslant:} x.\ \sigma$) bounds.

Future work

- Wehr and Thiemann [2009] reduce $F^D_{\leqslant:}$ subtyping to subtyping existential types with upper ($\exists x {\leqslant:} \tau.\ \sigma$) and lower ($\exists \tau {\leqslant:} x.\ \sigma$) bounds.
  Incomplete mechanization; syntax has types *and classes*.

Future work

- Wehr and Thiemann [2009] reduce $F^D_{\leqslant:}$ subtyping to subtyping existential types with upper ($\exists x {\leqslant:} \tau.\ \sigma$) and lower ($\exists \tau {\leqslant:} x.\ \sigma$) bounds.
  Incomplete mechanization; syntax has types *and classes.*
- Hu and Lhoták [2020] reduce $F^N_{\leqslant:}$ subtyping to subtyping Dependent-Object types (the core calculus of Scala).
  Not ported to Coq due to time constraints.

Bibliography

Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys (CSUR)*, 17(4):471–523, 1985. ISSN 15577341. doi: 10.1145/6041.6042. URL https://dl.acm.org/doi/10.1145/6041.6042.

Giorgio Ghelli. *Proof Theoretic Studies about a Minimal Type System Integrating Inclusion and Parametric Polymorphism*. PhD thesis, 1990.

Giorgio Ghelli. Divergence of $F_{\leqslant:}$ type checking. *Theoretical Computer Science*, 139(1-2): 131–162, 1995. ISSN 03043975. doi: 10.1016/0304-3975(94)00037-J. URL https://linkinghub.elsevier.com/retrieve/pii/030439759400037J.

Benjamin Pierce. Bounded Quantification Is Undecidable. *Information and Computation*, 112(1):131–165, jul 1994. ISSN 08905401. doi: 10.1006/inco.1994.1055. URL https://linkinghub.elsevier.com/retrieve/pii/S0890540184710558.

Stefan Wehr and Peter Thiemann. On the decidability of subtyping with bounded existential types. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5904 LNCS, pages 111–127, 2009. ISBN 3642106714. doi: 10.1007/978-3-642-10672-9_10. URL http://link.springer.com/10.1007/978-3-642-10672-9_10.

Jason Hu and Ondrej Lhoták. Undecidability of $D_{\leqslant:}$ and Its Decidable Fragments. *Proceedings of the ACM on Programming Languages*, 4(POPL), 2020. doi: 10.1145/3371077. URL https://doi.org/10.1145/3371077.