Practical Curry-Style using Choice Operators, Local Subtyping and Circular Proofs

Rodolphe Lepigre, Christophe Raffalli

Our goal: a classical, Curry-style proof system

PML_2: ML-like language with support for proofs of programs

Non-exhaustive list of features:

- Sum types (variants) and product types (records)
- Recursion, effects, call-by-value evaluation
- Curry-style quantifiers (polymorphism, type abstraction),
- Inductive and coinductive types (with sizes)
- Termination checking (only required for proofs)
- Untyped terms as the individuals of the underlying logic
- Restriction types $A \upharpoonright P$ with a “semantic predicate” P
- Membership type $t \in A$ used to encode dependent types

Our goal: a classical, Curry-style proof system

PML_2: ML-like language with support for proofs of programs

Non-exhaustive list of features:

- Sum types (variants) and product types (records)
- Recursion, effects, call-by-value evaluation
- Curry-style quantifiers (polymorphism, type abstraction),
- Inductive and coinductive types (with sizes)
- Termination checking (only required for proofs)
- Untyped terms as the individuals of the underlying logic
- Restriction types $A \upharpoonright P$ with a "semantic predicate" P
- Membership type $t \in A$ used to encode dependent types

~~PML_2: ML-like language with support for proofs of programs~~

Non-exhaustive list of features:

- Sum types (variants) and product types (records)
- Recursion, effects, call-by-value evaluation
- Curry-style quantifiers (polymorphism, type abstraction),
- Inductive and coinductive types (with sizes)
- Termination checking (only required for proofs)
- Untyped terms as the individuals of the underlying logic
- Restriction types $A \upharpoonright P$ with a "semantic predicate" P
- Membership type $t \in A$ used to encode dependent types

System F à la Church

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B}$$

$$\frac{\Gamma \vdash \color{red}{t} : A \quad X \notin \Gamma}{\Gamma \vdash \color{red}{\Lambda X.t} : \forall X.A} \qquad \frac{\Gamma \vdash \color{red}{t} : \forall X.A}{\Gamma \vdash \color{red}{t\ B} : A[X := B]}$$

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B}$$

$$\frac{\Gamma \vdash \color{red}{t} : A \qquad X \notin \Gamma}{\Gamma \vdash \color{red}{t} : \forall X.A}$$

$$\frac{\Gamma \vdash \color{red}{t} : \forall X.A}{\Gamma \vdash \color{red}{t} : A[X := B]}$$

How can we make Curry-style practical?

Typability and Type-Checking in the Second-Order lambda-Calculus are Equivalent and Undecidable (Joe B. Wells, LICS 1994)

Is that really a problem?

How can we make Curry-style practical?

Typability and Type-Checking in the Second-Order lambda-Calculus are Equivalent and Undecidable (Joe B. Wells, LICS 1994)

Is that really a problem? No!

How can we implement rules that are not syntax-directed?

How can we make Curry-style practical?

Typability and Type-Checking in the Second-Order lambda-Calculus are Equivalent and Undecidable (Joe B. Wells, LICS 1994)

Is that really a problem? No!

How can we implement rules that are not syntax-directed? We don't!

Main idea: completely reformulate the system using subtyping

Our extension of System F has many forms of subtyping:

- On quantifiers $\forall X.(A \Rightarrow B) \subseteq (\forall X.A) \Rightarrow (\forall X.B)$
- On sums (or variants) $[T|F] \subseteq [T|F|M]$
- On products (records) $\{x : R\,;\, y : R\,;\, z : R\} \subseteq \{x : R\,;\, y : R\}$
- On (sized) inductive types $\mu_\tau X.A \subseteq \mu_\kappa X.A$ (when $\tau \leqslant \kappa$)
- And similarly on (sized) coinductive types

Many different forms of subtyping

Our extension of System F has many forms of subtyping:

- On quantifiers $\forall X.(A \Rightarrow B) \subseteq (\forall X.A) \Rightarrow (\forall X.B)$
- On sums (or variants) $[T|F] \subseteq [T|F|M]$
- On products (records) $\{x : R ; y : R ; z : R\} \subseteq \{x : R ; y : R\}$
- On (sized) inductive types $\mu_\tau X.A \subseteq \mu_\kappa X.A$ (when $\tau \leqslant \kappa$)
- And similarly on (sized) coinductive types

Remark: PML_2 also has $(A \restriction P) \subseteq A$ and $(t \in A) \subseteq A$

Part I Going syntax-directed with local subtyping

Part II Choice operators and semantics

Part III Sized types, circular proofs and termination

Part I

Going syntax-directed with local subtyping

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash t : A \qquad X \notin \Gamma}{\Gamma \vdash t : \forall X.A} \qquad \frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t : A[X := B]}$$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}$$

$$\frac{\Gamma \vdash t : A \qquad X \notin \Gamma}{\Gamma \vdash t : \forall X.A} \qquad \frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t : A[X := B]}$$

Remarks on what needs to be changed:

- Quantifier rules definitely need to go

Toward syntax-directed typing rules

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash t : A \quad X \notin \Gamma}{\Gamma \vdash t : \forall X.A} \qquad \frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t : A[X := B]}$$

Remarks on what needs to be changed:

- Quantifier rules definitely need to go
- Arrow introduction is too restrictive (only function type)

Toward syntax-directed typing rules

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash t : A \qquad X \notin \Gamma}{\Gamma \vdash t : \forall X.A} \qquad \frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t : A[X := B]}$$

Remarks on what needs to be changed:

- Quantifier rules definitely need to go
- Arrow introduction is too restrictive (only function type)
- Arrow elimination rule is fine (no assumption on B)

Revisiting arrow introduction: naive subtyping

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

We should be able to prove judgments of the form $\Gamma \vdash \lambda x.t : \forall X.C$

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

We should be able to prove judgments of the form $\Gamma \vdash \lambda x.t : \forall X.C$

What about the following rule?

$$\frac{A \Rightarrow B \subseteq C \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

Revisiting arrow introduction: naive subtyping

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

We should be able to prove judgments of the form $\Gamma \vdash \lambda x.t : \forall X.C$

What about the following rule?

$$\frac{A \Rightarrow B \subseteq C \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

Not good enough (eigenvariable constraint not expressible)

$$\frac{\Gamma \vdash t : A \quad X \notin \Gamma}{\Gamma \vdash t : \forall X.A}$$

We may rather rely on a form of "judgment implication"

$$\frac{(\Gamma \vdash \lambda x.t : A \Rightarrow B) \subseteq (\Gamma \vdash \lambda x.t : C) \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

We may rather rely on a form of "judgment implication"

$$\frac{(\Gamma \vdash \lambda x.t : A \Rightarrow B) \subseteq (\Gamma \vdash \lambda x.t : C) \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

Example:

$$\dfrac{\dfrac{\overline{(\Gamma \vdash \lambda x.x : X \Rightarrow X) \subseteq (\Gamma \vdash \lambda x.x : X \Rightarrow X)} \qquad X \notin \Gamma}{(\Gamma \vdash \lambda x.x : X \Rightarrow X) \subseteq (\Gamma \vdash \lambda x.x : \forall X.X \Rightarrow X)} \qquad \overline{\Gamma, x : X \vdash x : X}}{\Gamma \vdash \lambda x.x : \forall X.X \Rightarrow X}$$

After removing the redundant information we get the rule

$$\frac{\Gamma \vdash \lambda x.t : A \Rightarrow B \subseteq C \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

After removing the redundant information we get the rule

$$\frac{\Gamma \vdash \lambda x.t : A \Rightarrow B \subseteq C \qquad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

The previous example then becomes

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \lambda x.x : X \Rightarrow X \subseteq X \Rightarrow X} \qquad X \notin \Gamma}{\Gamma \vdash \lambda x.x : X \Rightarrow X \subseteq \forall X.X \Rightarrow X} \qquad \dfrac{}{\Gamma, x : X \vdash x : X}}{\Gamma \vdash \lambda x.x : \forall X.X \Rightarrow X}$$

Type system for (Curry-style) System F

$$\frac{\Gamma, x : A \vdash x : A \subseteq B}{\Gamma, x : A \vdash x : B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash \lambda x.t : A \Rightarrow B \subseteq C \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : C}$$

$$\frac{}{\Gamma \vdash t : A \subseteq A} \qquad \frac{\Gamma, x : C \vdash x : C \subseteq A \quad \Gamma, x : C \vdash t\ x : B \subseteq D}{\Gamma \vdash t : A \Rightarrow B \subseteq C \Rightarrow D}$$

$$\frac{\Gamma \vdash t : A \subseteq B \quad X \notin \Gamma}{\Gamma \vdash t : A \subseteq \forall X.B} \qquad \frac{\Gamma \vdash t : A[X := C] \subseteq B}{\Gamma \vdash t : \forall X.A \subseteq B}$$

Can be implemented with standard unification variables

$$\frac{\Gamma, x : A \vdash x : A \subseteq B}{\Gamma, x : A \vdash x : B} \qquad \frac{\Gamma \vdash t : U \Rightarrow B \quad \Gamma \vdash u : U}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma \vdash \lambda x.t : U \Rightarrow V \subseteq C \quad \Gamma, x : U \vdash t : V}{\Gamma \vdash \lambda x.t : C}$$

$$\frac{A = B}{\Gamma \vdash t : A \subseteq B} \qquad \frac{\Gamma, x : C \vdash x : C \subseteq A \quad \Gamma, x : C \vdash t\ x : B \subseteq D}{\Gamma \vdash t : A \Rightarrow B \subseteq C \Rightarrow D}$$

$$\frac{\Gamma \vdash t : A \subseteq B \quad X \notin \Gamma}{\Gamma \vdash t : A \subseteq \forall X.B} \qquad \frac{\Gamma \vdash t : A[X := U] \subseteq B}{\Gamma \vdash t : \forall X.A \subseteq B}$$

Part II

Choice operators and semantics

We use "symbols" like $\varepsilon_{x\in A}(t \notin B)$ or $\varepsilon_X(t \notin A)$ instead of free variables

$$t, u ::= x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x\in A}(t \notin B) \qquad (\Lambda)$$

$$A, B ::= X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) \qquad (\mathcal{F})$$

We use "symbols" like $\varepsilon_{x \in A}(t \notin B)$ or $\varepsilon_X(t \notin A)$ instead of free variables

$$\begin{aligned} t, u &::= x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x\in A}(t \notin B) && (\Lambda) \\ A, B &::= X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) && (\mathcal{F}) \end{aligned}$$

Choice operators carry a representation of their semantics:

- $\varepsilon_{x\in A}(t \notin B)$ denotes "a term u in A such that $t[x := u]$ is not in B"
- $\varepsilon_X(t \notin A)$ denotes "a type C such that t is not in $A[X := C]$"

We use "symbols" like $\varepsilon_{x \in A}(t \notin B)$ or $\varepsilon_X(t \notin A)$ instead of free variables

$$t, u ::= x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x \in A}(t \notin B) \qquad (\Lambda)$$
$$A, B ::= X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) \qquad (\mathcal{F})$$

Choice operators carry a representation of their semantics:

- $\varepsilon_{x \in A}(t \notin B)$ denotes "a term u in A such that $t[x := u]$ is not in B"
- $\varepsilon_X(t \notin A)$ denotes "a type C such that t is not in $A[X := C]$"

Remark: in $\varepsilon_{x \in A}(t \notin B)$ we enforce $FV(t) \subseteq \{x\}$

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $[\![\Lambda]\!] = \{t \in \Lambda \mid t$ contains no "ε"$\}$
- $[\![\mathcal{F}]\!] = \{\Phi \subseteq [\![\Lambda]\!] \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N}$ and Φ is "saturated"$\}$

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $[\![\Lambda]\!] = \{t \in \Lambda \mid t$ contains no "ε"$\}$
- $[\![\mathcal{F}]\!] = \{\Phi \subseteq [\![\Lambda]\!] \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N}$ and Φ is "saturated"$\}$

Theorem (adequacy):

- if $t : A$ is derivable then $[\![t]\!] \in [\![A]\!]$
- if $t : A \subseteq B$ is derivable and $[\![t]\!] \in [\![A]\!]$ then $[\![t]\!] \in [\![B]\!]$

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $[\![\Lambda]\!] = \{t \in \Lambda \mid t \text{ contains no "}\varepsilon\text{"}\}$
- $[\![\mathcal{F}]\!] = \{\Phi \subseteq [\![\Lambda]\!] \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N} \text{ and } \Phi \text{ is "saturated"}\}$

Theorem (adequacy):

- if $t : A$ is derivable then $[\![t]\!] \in [\![A]\!]$
- if $t : A \subseteq B$ is derivable and $[\![t]\!] \in [\![A]\!]$ then $[\![t]\!] \in [\![B]\!]$

Proof: by induction on the typing / subtyping derivation

Definition of the interpretation fonctions

$[\![-]\!] : \Lambda \to [\![\Lambda]\!]$ is defined as:

$$[\![x]\!] = x \qquad [\![\lambda x.t]\!] = \lambda x.[\![t]\!] \qquad [\![t\ u]\!] = [\![t]\!][\![u]\!]$$

$$[\![\varepsilon_{x \in A}(t \notin B)]\!] = u \in [\![A]\!], \text{ such that } [\![t[x:=u]]\!] \notin [\![B]\!] \text{ if possible}$$

Definition of the interpretation fonctions

$[\![-]\!] : \Lambda \to [\![\Lambda]\!]$ is defined as:

$$[\![x]\!] = x \qquad [\![\lambda x.t]\!] = \lambda x.[\![t]\!] \qquad [\![t\ u]\!] = [\![t]\!][\![u]\!]$$

$$[\![\varepsilon_{x\in A}(t \notin B)]\!] = u \in [\![A]\!], \text{ such that } [\![t[x:=u]]\!] \notin [\![B]\!] \text{ if possible}$$

$[\![-]\!] : \mathcal{F} \to [\![\mathcal{F}]\!]$ is defined as:

$$[\![A \Rightarrow B]\!] = \{t \in [\![\Lambda]\!] \mid \forall u \in [\![A]\!], t\ u \in [\![B]\!]\}$$

$$[\![\forall X.A]\!] = \bigcap_{\Phi \in [\![\mathcal{F}]\!]} [\![A[X := \Phi]]\!] \qquad [\![\Phi]\!] = \Phi$$

$$[\![\varepsilon_X(t \notin A)]\!] = \Phi \in [\![\mathcal{F}]\!], \text{ such that } [\![t]\!] \notin [\![A[X := \Phi]]\!] \text{ if possible}$$

Examples of adequate typing and subtyping rules

$$\frac{\lambda x.t : A \Rightarrow B \subseteq C \qquad t[x := \varepsilon_{x \in A}(t \notin B)] : B}{\lambda x.t : C}$$

Examples of adequate typing and subtyping rules

$$\frac{\lambda x.t : A \Rightarrow B \subseteq C \qquad t[x := \varepsilon_{x \in A}(t \notin B)] : B}{\lambda x.t : C}$$

$$\frac{\varepsilon_{x \in A}(t \notin B) : A \subseteq C}{\varepsilon_{x \in A}(t \notin B) : C}$$

Examples of adequate typing and subtyping rules

$$\frac{\lambda x.t : A \Rightarrow B \subseteq C \qquad t[x := \varepsilon_{x \in A}(t \notin B)] : B}{\lambda x.t : C}$$

$$\frac{\varepsilon_{x \in A}(t \notin B) : A \subseteq C}{\varepsilon_{x \in A}(t \notin B) : C}$$

$$\frac{t : A[X := C] \subseteq B}{t : \forall X.A \subseteq B}$$

$$\frac{\varepsilon_{x\in A}(t \notin B) : A \subseteq C}{\varepsilon_{x\in A}(t \notin B) : C} \qquad \frac{t : A \Rightarrow B \quad u : A}{t\ u : B}$$

$$\frac{\lambda x.t : A \Rightarrow B \subseteq C \quad t[x := \varepsilon_{x\in A}(t \notin B)] : B}{\lambda x.t : C}$$

$$\frac{}{t : A \subseteq A} \qquad \frac{\varepsilon_{x\in C}(t\ x \notin D) : C \subseteq A \quad t\ \varepsilon_{x\in C}(t\ x \notin D) : B \subseteq D}{t : A \Rightarrow B \subseteq C \Rightarrow D}$$

$$\frac{t : A \subseteq B[X := \varepsilon_X(t \notin B)]}{t : A \subseteq \forall X.B} \qquad \frac{t : A[X := C] \subseteq B}{t : \forall X.A \subseteq B}$$

Can still be implemented with standard unification variables

$$\frac{\varepsilon_{x\in A}(t \notin B) : A \subseteq C}{\varepsilon_{x\in A}(t \notin B) : C} \qquad \frac{t : U \Rightarrow B \quad u : U}{t\ u : B}$$

$$\frac{\lambda x.t : U \Rightarrow V \subseteq C \quad t[x := \varepsilon_{x\in U}(t \notin V)] : V}{\lambda x.t : C}$$

$$\frac{A = B}{t : A \subseteq B} \qquad \frac{\varepsilon_{x\in C}(t\ x \notin D) : C \subseteq A \quad t\ \varepsilon_{x\in C}(t\ x \notin D) : B \subseteq D}{t : A \Rightarrow B \subseteq C \Rightarrow D}$$

$$\frac{t : A \subseteq B[X := \varepsilon_X(t \notin B)]}{t : A \subseteq \forall X.B} \qquad \frac{t : A[X := U] \subseteq B}{t : \forall X.A \subseteq B}$$

Why reformulate the system with choice operators?

Advantages of this presentation:

- Maximal weakening is applied automatically
- Judgments easier to recognise when building circular proofs
- The "free variables" carry their semantics

Why reformulate the system with choice operators?

Advantages of this presentation:

- Maximal weakening is applied automatically
- Judgments easier to recognise when building circular proofs
- The "free variables" carry their semantics

Only one drawback:

- Terms may become very big

Part III

Sized types, circular proofs, and termination

We add a new type former to the system:

$$
\begin{aligned}
t, u &::= x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x \in A}(t \notin B) && (\Lambda)\\
A, B &::= X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) \mid \mu_\tau X.A && (\mathcal{F})\\
\tau, \kappa &::= \alpha \mid \kappa+1 \mid \infty \mid \varepsilon_{\alpha<\kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) && (\mathcal{O})
\end{aligned}
$$

Fixpoints are indexed with “syntactic ordinals” (interpreted as ordinals)

We add a new type former to the system:

$$
\begin{array}{rcll}
t, u & ::= & x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x \in A}(t \notin B) & (\Lambda) \\
A, B & ::= & X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) \mid \mu_\tau X.A & (\mathcal{F}) \\
\tau, \kappa & ::= & \alpha \mid \kappa + 1 \mid \infty \mid \varepsilon_{\alpha < \kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) & (\mathcal{O})
\end{array}
$$

Fixpoints are indexed with "syntactic ordinals" (interpreted as ordinals)

$$
[\![\mu_\tau X.A]\!] = \bigcup_{o < [\![\tau]\!]} [\![A[X := \mu_o X.A]]\!]
$$

We add a new type former to the system:

$$
\begin{aligned}
t, u &::= x \mid \lambda x.t \mid t\ u \mid \varepsilon_{x \in A}(t \notin B) && (\Lambda)\\
A, B &::= X \mid A \Rightarrow B \mid \forall X.A \mid \varepsilon_X(t \notin A) \mid \mu_\tau X.A && (\mathcal{F})\\
\tau, \kappa &::= \alpha \mid \kappa + 1 \mid \infty \mid \varepsilon_{\alpha < \kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) && (\mathcal{O})
\end{aligned}
$$

Fixpoints are indexed with "syntactic ordinals" (interpreted as ordinals)

$$
[\![\mu_\tau X.A]\!] = \bigcup_{o < [\![\tau]\!]} [\![A[X := \mu_o X.A]]\!]
$$

$[\![\kappa + 1]\!] = [\![\kappa]\!] + 1$ $\qquad$ $[\![\infty]\!] =$ "a large enough ordinal" $\qquad$ ...

Adequate rules for the least fixpoint constructor:

$$\frac{t : A \subseteq B[X{:=}\mu_\infty X.B]}{t : A \subseteq \mu_\infty X.B} \qquad \frac{t : A \subseteq B[X{:=}\mu_\tau X.B] \quad \tau < \kappa}{t : A \subseteq \mu_\kappa X.B}$$

$$\frac{t : A[X{:=}\mu_\tau X.A] \subseteq B}{t : \mu_\kappa X.A \subseteq B} \quad \text{with} \quad \tau = \varepsilon_{\alpha<\kappa}(t \in A[X{:=}\mu_\alpha X.A])$$

Adequate rules for the least fixpoint constructor:

$$\frac{t : A \subseteq B[X := \mu_\infty X.B]}{t : A \subseteq \mu_\infty X.B} \qquad \frac{t : A \subseteq B[X := \mu_\tau X.B] \quad \tau < \kappa}{t : A \subseteq \mu_\kappa X.B}$$

$$\frac{t : A[X := \mu_\tau X.A] \subseteq B}{t : \mu_\kappa X.A \subseteq B} \quad \text{with} \quad \tau = \varepsilon_{\alpha < \kappa}(t \in A[X := \mu_\alpha X.A])$$

Question: when do we stop the unfolding?

Introducing a cyclic structure (generalisation, induction)

$$\frac{A \subseteq B}{t : A \subseteq B} \qquad\qquad \frac{\varepsilon_{x \in A}(x \notin B) : A \subseteq B}{A \subseteq B}$$

Introducing a cyclic structure (generalisation, induction)

$$\frac{A \subseteq B}{t : A \subseteq B} \qquad \frac{\varepsilon_{x \in A}(x \notin B) : A \subseteq B}{A \subseteq B}$$

$$\frac{\forall \alpha \ (A \subseteq B)}{A[\alpha := \kappa] \subseteq B[\alpha := \kappa]}$$

Introducing a cyclic structure (generalisation, induction)

$$\frac{A \subseteq B}{t : A \subseteq B} \qquad\qquad \frac{\varepsilon_{x \in A}(x \notin B) : A \subseteq B}{A \subseteq B}$$

$$\frac{\forall \alpha \ (A \subseteq B)}{A[\alpha := \kappa] \subseteq B[\alpha := \kappa]}$$

$$\dfrac{\dfrac{\overline{[\forall \alpha \ (A \subseteq B)]^{i}}}{\vdots}}{\dfrac{A[\alpha := \varepsilon_{\alpha}(A \not\subseteq B)] \subseteq B[\alpha := \varepsilon_{\alpha}(A \not\subseteq B)]}{\forall \alpha \ (A \subseteq B)}\, i}$$

We type the fixpoint combinator with a simple unrolling

$$\frac{t\,(\mathrm{fix}\ t) : A}{\mathrm{fix}\ t : A}$$

And we allow circularity in typing proofs (as with subtyping proofs)

$$\frac{\forall\alpha\ (t : A)}{t : A[\alpha := \kappa]} \qquad \dfrac{\dfrac{\overline{[\forall\alpha\ (t : A)]^i}}{\vdots}}{\dfrac{t : A[\alpha := \varepsilon_\alpha(t \notin A)]}{\forall\alpha\ (t : A)}\,i}$$

References for Technical Details

Practical Subtyping for System F with Sized (Co-)Induction

R. Lepigre and C. Raffalli, under revision

https://lepigre.fr/files/docs/lepigre2017_subml.pdf

https://github.com/rlepigre/subml

https://rlepigre.github.io/subml

Semantics and Implementation of an Extension of ML for Proving Programs

R. Lepigre, PhD manuscript

https://github.com/rlepigre/phd

https://github.com/rlepigre/pml

Thanks!