

PRACTICAL CURRY-STYLE USING CHOICE OPERATORS, LOCAL SUBTYPING AND CIRCULAR PROOFS



RODOLPHE LEPIGRE, CHRISTOPHE RAFFALLI

OUR GOAL: A CLASSICAL, CURRY-STYLE PROOF SYSTEM

PML₂: 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

PML₂: 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

~~PML₂: 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

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

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

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

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

SYSTEM F À LA CURRY

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

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

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

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

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

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 $\tau \leq \kappa$)
- 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 $\tau \leq \kappa$)
- And similarly on (sized) coinductive types

Remark: PML_2 also has $(A \upharpoonright P) \subseteq A$ 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

TOWARD SYNTAX-DIRECTED TYPING RULES

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

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

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

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

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

TOWARD SYNTAX-DIRECTED TYPING RULES

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

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

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

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

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

Remarks on what needs to be changed:

- Quantifier rules definitely need to go

TOWARD SYNTAX-DIRECTED TYPING RULES

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

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

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

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

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

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

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

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 $\Gamma \vdash \lambda x.t : \forall X.C$

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?

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

REVISITING ARROW INTRODUCTION: NAIVE SUBTYPING

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

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)

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

REVISITING ARROW INTRODUCTION: JUDGMENT IMPLICATION

We may rather rely on a form of “judgment implication”

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

REVISITING ARROW INTRODUCTION: JUDGMENT IMPLICATION

We may rather rely on a form of “judgment implication”

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

Example:

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

REVISITING ARROW INTRODUCTION: LOCAL SUBTYPING

After removing the redundant information we get the rule

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

REVISITING ARROW INTRODUCTION: LOCAL SUBTYPING

After removing the redundant information we get the rule

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

The previous example then becomes

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

TYPE SYSTEM FOR (CURRY-STYLE) SYSTEM F

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

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

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

$$\frac{}{\Gamma \vdash t:A \subseteq A}$$

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

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

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

CAN BE IMPLEMENTED WITH STANDARD UNIFICATION VARIABLES

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

$$\frac{\Gamma \vdash t : \mathbf{U} \Rightarrow B \quad \Gamma \vdash u : \mathbf{U}}{\Gamma \vdash t u : B}$$

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

$$\frac{\mathbf{A} = \mathbf{B}}{\Gamma \vdash t : A \subseteq B}$$

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

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

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

PART II

CHOICE OPERATORS AND SEMANTICS

ANOTHER FORMULATION OF THE SYSTEM USING CHOICE OPERATORS

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

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

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

ANOTHER FORMULATION OF THE SYSTEM USING CHOICE OPERATORS

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

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

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

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 $A[X := C]$ ”

ANOTHER FORMULATION OF THE SYSTEM USING CHOICE OPERATORS

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

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

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

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 $FV(t) \subseteq \{x\}$

REDUCIBILITY CANDIDATE SEMANTICS AND ADEQUACY

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $\llbracket \Lambda \rrbracket = \{t \in \Lambda \mid t \text{ contains no “}\varepsilon\text{”}\}$
- $\llbracket \mathcal{F} \rrbracket = \{\Phi \subseteq \llbracket \Lambda \rrbracket \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N} \text{ and } \Phi \text{ is “saturated”}\}$

REDUCIBILITY CANDIDATE SEMANTICS AND ADEQUACY

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $\llbracket \Lambda \rrbracket = \{t \in \Lambda \mid t \text{ contains no “}\varepsilon\text{”}\}$
- $\llbracket \mathcal{F} \rrbracket = \{\Phi \subseteq \llbracket \Lambda \rrbracket \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N} \text{ and } \Phi \text{ is “saturated”}\}$

Theorem (adequacy):

- if $t : A$ is derivable then $\llbracket t \rrbracket \in \llbracket A \rrbracket$
- if $t : A \subseteq B$ is derivable and $\llbracket t \rrbracket \in \llbracket A \rrbracket$ then $\llbracket t \rrbracket \in \llbracket B \rrbracket$

REDUCIBILITY CANDIDATE SEMANTICS AND ADEQUACY

We interpret terms as well as types

The domains of interpretation for terms and types are:

- $\llbracket \Lambda \rrbracket = \{t \in \Lambda \mid t \text{ contains no “}\varepsilon\text{”}\}$
- $\llbracket \mathcal{F} \rrbracket = \{\Phi \subseteq \llbracket \Lambda \rrbracket \mid \mathcal{N}_0 \subseteq \Phi \subseteq \mathcal{N} \text{ and } \Phi \text{ is “saturated”}\}$

Theorem (adequacy):

- if $t : A$ is derivable then $\llbracket t \rrbracket \in \llbracket A \rrbracket$
- if $t : A \subseteq B$ is derivable and $\llbracket t \rrbracket \in \llbracket A \rrbracket$ then $\llbracket t \rrbracket \in \llbracket B \rrbracket$

Proof: by induction on the typing / subtyping derivation

DEFINITION OF THE INTERPRETATION FONCTIONS

$\llbracket - \rrbracket : \Lambda \rightarrow \llbracket \Lambda \rrbracket$ is defined as:

$$\llbracket x \rrbracket = x \qquad \llbracket \lambda x. t \rrbracket = \lambda x. \llbracket t \rrbracket \qquad \llbracket t \ u \rrbracket = \llbracket t \rrbracket \llbracket u \rrbracket$$

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

DEFINITION OF THE INTERPRETATION FONCTIONS

$\llbracket - \rrbracket : \Lambda \rightarrow \llbracket \Lambda \rrbracket$ is defined as:

$$\llbracket x \rrbracket = x \qquad \llbracket \lambda x. t \rrbracket = \lambda x. \llbracket t \rrbracket \qquad \llbracket t \ u \rrbracket = \llbracket t \rrbracket \llbracket u \rrbracket$$

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

$\llbracket - \rrbracket : \mathcal{F} \rightarrow \llbracket \mathcal{F} \rrbracket$ is defined as:

$$\llbracket A \Rightarrow B \rrbracket = \{ t \in \llbracket \Lambda \rrbracket \mid \forall u \in \llbracket A \rrbracket, t \ u \in \llbracket B \rrbracket \}$$

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

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

EXAMPLES OF ADEQUATE TYPING AND SUBTYPING RULES

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

EXAMPLES OF ADEQUATE TYPING AND SUBTYPING RULES

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

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

EXAMPLES OF ADEQUATE TYPING AND SUBTYPING RULES

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

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

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

TYPE SYSTEM FOR (CURRY-STYLE) SYSTEM F (ε VERSION)

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

$$\frac{t : A \Rightarrow B \quad u : A}{t \ u : B}$$

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

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

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

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

CAN STILL BE IMPLEMENTED WITH STANDARD UNIFICATION VARIABLES

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

$$\frac{t : \mathbf{U} \Rightarrow B \quad u : \mathbf{U}}{t \ u : B}$$

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

$$\frac{\mathbf{A} = \mathbf{B}}{t : A \subseteq B}$$

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

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

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

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

EXTENDING THE SYSTEM WITH (SIZED) INDUCTIVE TYPES

We add a new type former to the system:

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

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

$$\tau, \kappa ::= \alpha \mid \kappa + 1 \mid \infty \mid \varepsilon_{\alpha < \kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) \quad (\mathcal{O})$$

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

EXTENDING THE SYSTEM WITH (SIZED) INDUCTIVE TYPES

We add a new type former to the system:

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

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

$$\tau, \kappa ::= \alpha \mid \kappa + 1 \mid \infty \mid \varepsilon_{\alpha < \kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) \quad (\mathcal{O})$$

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

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

EXTENDING THE SYSTEM WITH (SIZED) INDUCTIVE TYPES

We add a new type former to the system:

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

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

$$\tau, \kappa ::= \alpha \mid \kappa + 1 \mid \infty \mid \varepsilon_{\alpha < \kappa}(t \in A) \mid \varepsilon_\alpha(A \not\subseteq B) \quad (\mathcal{O})$$

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

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

$$\llbracket \kappa + 1 \rrbracket = \llbracket \kappa \rrbracket + 1 \qquad \llbracket \infty \rrbracket = \text{“a large enough ordinal”} \qquad \dots$$

SUBTYPING RULES FOR INDUCTIVE TYPES

Adequate rules for the least fixpoint constructor:

$$\frac{t : A \subseteq B[X := \mu_{\infty} X.B]}{t : A \subseteq \mu_{\infty} X.B}$$

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

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

SUBTYPING RULES FOR INDUCTIVE TYPES

Adequate rules for the least fixpoint constructor:

$$\frac{t : A \subseteq B[X := \mu_{\infty} X.B]}{t : A \subseteq \mu_{\infty} X.B}$$

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

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

Question: when do we stop the unfolding?

INTRODUCING A CYCLIC STRUCTURE (GENERALISATION, INDUCTION)

$$\frac{A \subseteq B}{t : A \subseteq B}$$

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

INTRODUCING A CYCLIC STRUCTURE (GENERALISATION, INDUCTION)

$$\frac{A \subseteq B}{t : A \subseteq B}$$

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

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

INTRODUCING A CYCLIC STRUCTURE (GENERALISATION, INDUCTION)

$$\frac{A \subseteq B}{t : A \subseteq B}$$

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

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

$$\frac{\begin{array}{c} [\forall \alpha (A \subseteq B)]^i \\ \vdots \end{array}}{\frac{A[\alpha := \varepsilon_\alpha(A \not\subseteq B)] \subseteq B[\alpha := \varepsilon_\alpha(A \not\subseteq B)]}{\forall \alpha (A \subseteq B)}_i}$$

FIXPOINT COMBINATOR AND RECURSION

We type the fixpoint combinator with a simple unrolling

$$\frac{t(\text{fix } t) : A}{\text{fix } t : A}$$

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

$$\frac{\forall \alpha \ (t : A)}{t : A[\alpha := \kappa]} \qquad \frac{\begin{array}{c} [\forall \alpha \ (t : A)]^i \\ \vdots \\ t : A[\alpha := \varepsilon_\alpha(t \notin A)] \end{array}}{\forall \alpha \ (t : A)}_i$$

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!