A Practical Framework for Curry-Style Languages
(Inspired by realizability semantics)

Rodolphe Lepigre
Context: using realizability for programming languages

Last year’s talk was about the PML language:
- A simple but powerful mechanism for program certification
- It is embedded in a (fairly standard) ML-style language
- Everything is backed by a (classical) realizability semantics
- Property: \( v \in \phi^\perp \Rightarrow v \in \phi \) for all \( \phi \) closed under \( (\equiv) \)

Today’s talk is about making Curry-style quantifiers practical:
- They are essential for PML (polymorphism, dependent types)
- But pose a practical issue due to non-syntax-directed rules
- Restricting quantifiers (prenex polymorphism) is not an option
- Contribution: a solution with subtyping inspired by semantics

In this talk we will stick to System F for simplicity
Quick reminder: Church-style versus Curry-style

Church-style System F:

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

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

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

Curry-style System F is obtained by removing the highlighted parts
A natural idea: using subtyping

We define a relation (∍) on types and use rule:

\[
\Gamma \vdash t : A \quad A \sqsubseteq B \\
\Gamma \vdash t : B 
\]

This does help a bit already:

\[
\begin{align*}
A \sqsubseteq C & \quad \Gamma, x : A \vdash x : C \\
A \Rightarrow B \subseteq C & \quad \Gamma, x : A \vdash t : B \\
\Gamma \vdash \lambda x. t : C \\
\Gamma \vdash t \ u : B
\end{align*}
\]

Ideally we would want quantifiers to be handled by subtyping
Containment system [Mitchell]

Is standard containment enough?

\[
\{ Y_1, \ldots, Y_m \} \cap FV(\forall X_1 \ldots \forall X_n.A) = \emptyset
\]

\[
\forall X_1 \ldots \forall X_n.A \subseteq \forall Y_1 \ldots \forall Y_m.A[X_1 := B_1, \ldots, X_n := B_n]
\]

\[
\forall X_1 \ldots \forall X_n.A \Rightarrow B \subseteq (\forall X_1 \ldots \forall X_n.A) \Rightarrow (\forall X_1 \ldots \forall X_n.B)
\]

\[
A_2 \subseteq A_1 \quad B_1 \subseteq B_2
\]

\[
A_1 \Rightarrow B_1 \subseteq A_2 \Rightarrow B_2
\]

\[
A \subseteq B \quad B \subseteq C
\]

\[
A \subseteq C
\]

\[
A \subseteq B
\]

\[
\forall X.A \subseteq \forall X.B
\]
Can we derive the quantifier rules?

Yes we can derive the elimination rule:

\[
\frac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t : A[X := B]} \triangleq \frac{\emptyset \cap \text{FV}(\forall X.A) = \emptyset}{\frac{\frac{\Gamma \vdash t : \forall X.A}{\forall X.A \subseteq A[X := B]} \quad \frac{\Gamma \vdash t : A[X := B]}{\Gamma \vdash t : A} }{\Gamma \vdash t : A[X := B]}
\]

No we cannot derive the introduction rule:

\[
\frac{\Gamma \vdash t : A \quad X \notin \Gamma}{\Gamma \vdash t : \forall X.A} \triangleq \frac{??}{\frac{\Gamma \vdash t : A \quad A \subseteq \forall X.A}{\Gamma \vdash t : \forall X.A}}
\]
Let us take a step back...

All we want is adequacy:

- If $\vdash t : A$ is derivable then $t \in \llbracket A \rrbracket$
- If $A \subseteq B$ then $\llbracket A \rrbracket \subseteq \llbracket B \rrbracket$

The subtyping part is not as fine-grained as it could be:

\[
\frac{\vdash t : A \quad A \subseteq B}{\vdash t : B} \quad \text{can be replaced by} \quad \frac{\vdash t : A \quad \vdash t : A \subseteq B}{\vdash t : B}
\]

Local subtyping is interpreted as an implication
Approach 1
(inspired by semantics)
Main idea of the approach

Based on a fine-grained semantic analysis we:

- Get rid of context and only work with closed terms
- To this aim terms are extended with choice operators
- The same kind of trick is used for quantifiers in types

Theorem (Adequacy)

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

Terms are interpreted using “pure terms"
(satisfying the intended semantic property)
Typing and subtyping rules

Syntax-directed typing rules:

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

\[ \frac{t : A \Rightarrow B \quad u : A}{t \ u : B} \]

\[ \frac{\lambda x.t : A \Rightarrow B \subseteq C}{\lambda x.t : C} \]

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

Syntax-directed (local) subtyping rules:

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

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

\[ \frac{\varepsilon_{x \in A_2}(t \notin B_2) : A_2 \subseteq A_1}{t \varepsilon_{x \in A_2}(t \notin B_2) : B_1 \subseteq B_2} \]

\[ \frac{t : A_1 \Rightarrow B_1 \subseteq A_2 \Rightarrow B_2}{t : A_1 \Rightarrow B_1 \subseteq A_2 \Rightarrow B_2} \]
Interpretation of terms and types

We interpret terms using "pure terms" (without choice operators)

\[ [x] = x \quad [\lambda x. t] = \lambda x. [t] \quad [t \ u] = [t] \ [u] \]

\[ [\exists x \in A (t^* \notin B)] = \begin{cases} \{ u \in [A] \text{ s.t. } [t[x := u]] \notin [B] \text{ if it exists} \\ \text{any } t \in \mathcal{N}_0 \text{ otherwise} \end{cases} \]

We interpret types as (saturated) sets of normalizing terms

\[ [\Phi] = \Phi \quad [A \rightarrow B] = [A] \rightarrow [B] \quad [\forall X. A] = \bigcap_{\Phi \in \mathcal{F}} [A[X := \Phi]] \]

\[ [\exists X (t \notin A)] = \begin{cases} \{ \Phi \in \mathcal{F} \text{ such that } [t] \notin [A[X := \Phi]] \text{ if it exists} \\ \mathcal{N}_0 \text{ otherwise} \end{cases} \]

\[ \Phi \Rightarrow \Psi = \{ t \mid \forall u \in \Phi, t \ u \in \Psi \} \]
Let us look at one case of the adequacy lemma

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

\[
[[\varepsilon_{x \in A}(t^* \notin B)]] = \begin{cases} 
  u \in [[A]] \text{ s.t. } [[t[x := u]]] \notin [[B]] & \text{if it exists} \\
  \text{any } t \in \mathcal{N}_0 & \text{otherwise}
\end{cases}
\]
Approach 2
(using syntactic translations)
A more standard type system

Syntax-directed typing rules:

\[
\frac{\Gamma, x : A \vdash x : A \subseteq C}{\Gamma, x : A \vdash x : C} \quad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t \ u : B}
\]

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

Syntax-directed (local) subtyping rules:

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

\[
\frac{\Gamma, x : A_2 \vdash x : A_2 \subseteq A_1 \quad \Gamma, x : A_2 \vdash t \ x : B_1 \subseteq B_2}{\Gamma \vdash t : A_1 \Rightarrow B_1 \subseteq A_2 \Rightarrow B_2}
\]
Elimination of subtyping: translation to System F+\(\eta\)

System F+\(\eta\) is obtained by adding the rule:

\[
\frac{\Gamma \vdash \lambda x.t \, x : A \Rightarrow B \quad x \notin t}{\Gamma \vdash t : A \Rightarrow B}
\]

Theorem (Translation to F+\(\eta\))

- If \(\Gamma \vdash t : A\) is derivable then it is also derivable in System F+\(\eta\)
- If \(\Gamma \vdash t : A \subseteq B\) is derivable then \(\Gamma \vdash t : B\) is derivable in System F+\(\eta\) given a derivation of \(\Gamma \vdash t : A\)

Translation of subtyping leads to a “piece of proof”:

\[
\begin{align*}
\Gamma \vdash t : A \\
\text{If } \Gamma \vdash t : A \subseteq B \text{ is derivable then we get} \\
\Pi \\
\Gamma \vdash t : B
\end{align*}
\]
The most interesting case (arrow subtyping rule)

\[
\Gamma, x : A_2 \vdash x : A_1 \quad \Gamma, x : A_2 \vdash t \; x : B_1 \subseteq B_2 \\
\Gamma \vdash t : A_1 \Rightarrow B_1 \subseteq A_2 \Rightarrow B_2
\]

\[
\Gamma \vdash t : A_1 \Rightarrow B_1 \\
\Gamma, x : A_2 \vdash t : A_1 \Rightarrow B_1 \quad x \text{ fresh} \\
\Gamma, x : A_2 \vdash x : A_1 \\
\Gamma, x : A_2 \vdash t \; x : B_1 \\
\Gamma, x : A_2 \vdash t \; x : B_2 \\
\Gamma \vdash \lambda x. t \; x : A_2 \Rightarrow B_2 \\
\Gamma \vdash t : A_2 \Rightarrow B_2
\]
Translation from System F+$\eta$

Given the subsumption rule the translation is immediate

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

A couple of remarks:

- We conjecture that subsumption is admissible
- The rule is useful anyway for ascription (rule below)
- (Remember that type-checking remains undecidable here)

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash t : A \subseteq B}{\Gamma \vdash (t : A) : B}$$
Thanks! Questions?

@ https://lepigre.fr
✉ lepigre@mpi-sws.org