System F with explicit type-level retyping functions

Didier Rémy

INRIA Paris - Rocquencourt

IFIP WG 2.8

Based on joint work with

Boris Yakobowski

An internal language for ML^{F}

This *is not* half-baked work

but

half of baked work

thus, I shall be short,
and I will not show you any graphs.

Prelude

About ML^F

ML^F smoothly combines System-F first-class polymorphism with ML-style type inference and requires very few type annotations—only on parameters of functions that are *used* polymorphically.

ML^F type inference crucially relies on type generalization for polymorphism explicit introduction, which is applied at every subexpression and not only at let-bindings. Therefore...

Prelude

About ML^F

ML^F smoothly combines System-F first-class polymorphism with ML-style type inference and requires very few type annotations—only on parameters of functions that are *used* polymorphically.

ML^F type inference crucially relies on type generalization for polymorphism explicit introduction, which is applied at every subexpression and not only at let-bindings. Therefore...

$$\mathrm{ML}^\mathsf{F} \heartsuit \mathrm{ML} \quad \wedge \quad \mathrm{ML} \heartsuit \mathrm{ML}^\mathsf{F}$$

Motivation

A fair question by Simon P.J. . . . in Island:

- What would be an internal language for ML^F ?

Type soundness for ML^F

- proved for the original version, but a bit tricky,
  using some intermediate more implicitly-typed version (no inference)
- it showed preservation of typability during reduction,
  but failed to preserve the typing derivation.
- it assumed that no reduction occurred under λ's.
- type soundness has never been proved for the most general version.

Also. . .

- Better understand *instance bounded quantification*,
  without having to bother about type inference.

The example of choice

$$
\begin{array}{lcl}
\text{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \text{if } \mathit{true} \text{ then } x \text{ else } y \\
 & & \\
\text{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x \\
 & & \\
\text{choice}\ \ \text{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & & \\
 & : & ?
\end{array}
$$

The example of choice

$$
\begin{array}{lrl}
\mathsf{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \mathsf{if}\ \mathit{true}\ \mathsf{then}\ x\ \mathsf{else}\ y
\end{array}
$$

$$
\begin{array}{lrl}
\mathsf{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}
$$

$$
\begin{array}{lrl}
\mathsf{choice}\ \ \mathsf{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta
\end{array}
$$

The example of choice

$$
\begin{array}{lll}
\text{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \text{if } \mathit{true} \text{ then } x \text{ else } y
\end{array}
$$

$$
\begin{array}{lll}
\text{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}
$$

$$
\begin{array}{lll}
\text{choice}\ \ \text{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta \\
 & \triangleq & ?
\end{array}
$$

The example of choice

$$\begin{array}{lll}
\mathsf{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
& \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \mathsf{if}\ \mathit{true}\ \mathsf{then}\ x\ \mathsf{else}\ y
\end{array}$$

$$\begin{array}{lll}
\mathsf{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
& \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}$$

$$\begin{array}{lll}
\mathsf{choice}\ \ \mathsf{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
& : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
& : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta
\end{array}$$

$$\triangleq \quad \Lambda(\beta \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \mathsf{choice}\ \square\ (\mathsf{id}\ \square)$$

The example of choice

$$
\begin{array}{lll}
\mathsf{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \mathsf{if}\ \mathit{true}\ \mathsf{then}\ x\ \mathsf{else}\ y
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{choice}\ \ \mathsf{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta
\end{array}
$$

$$
\triangleq \quad \Lambda(\beta \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \mathsf{choice}\ \langle\beta\rangle\ (\mathsf{id}\ (!\beta))
$$

The example of choice

$$
\begin{array}{lll}
\mathsf{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \mathsf{if}\ \mathit{true}\ \mathsf{then}\ x\ \mathsf{else}\ y
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{choice}\ \ \mathsf{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta
\end{array}
$$

$$
\triangleq \quad \Lambda(\beta \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \mathsf{choice}\ (\forall\,(\geqslant \beta); \&)\ \big(\mathsf{id}\ (!\beta)\big)
$$

The example of choice

$$
\begin{array}{lll}
\mathsf{choice} & : & \forall\,(\beta)\ \beta \to \beta \to \beta \\
 & \triangleq & \Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \mathsf{if}\ \mathit{true}\ \mathsf{then}\ x\ \mathsf{else}\ y
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{id} & : & \forall\,(\alpha)\ \alpha \to \alpha \\
 & \triangleq & \Lambda(\alpha)\ \lambda(x : \alpha)\ x
\end{array}
$$

$$
\begin{array}{lll}
\mathsf{choice}\ \ \mathsf{id} & : & (\forall\,(\alpha)\ \alpha \to \alpha) \to (\forall\,(\alpha)\ \alpha \to \alpha) \\
 & : & \forall\,(\alpha)\ (\alpha \to \alpha) \to (\alpha \to \alpha) \\
 & : & \forall\,(\alpha \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \beta \to \beta
\end{array}
$$

$$
\triangleq \quad \Lambda(\beta \geqslant \forall\,(\alpha)\ \alpha \to \alpha)\ \underbrace{\mathsf{choice}\ (\forall\,(\geqslant \beta); \&)}_{\beta \to \beta \to \beta}\ \big(\underbrace{\mathsf{id}\ (!\beta)}_{\beta}\big)
$$

Syntax

System F

Types

$$
\begin{array}{lll}
\tau & ::= & \\
 & \mid & \alpha \\
 & \mid & \tau \to \tau \\
 & \mid & \forall(\alpha) \, \tau \\
 & & \\
\phi & ::= & \tau
\end{array}
$$

Terms

$$
\begin{array}{lll}
a & ::= & \\
 & \mid & x \\
 & \mid & \lambda(x : \tau)\, a \\
 & \mid & a\ a \\
 & \mid & \Lambda(\alpha)\, a \\
 & \mid & a\,\phi \\
 & \mid & \mathsf{let}\ x = a\ \mathsf{in}\ a
\end{array}
$$

Syntax

ML^{F}

Types

$$
\begin{array}{lcl}
\tau & ::= & \\
 & | & \alpha \\
 & | & \tau \to \tau \\
 & | & \forall\,(\alpha \geqslant \tau)\ \tau \\
 & | & \bot \\
\phi & ::= & \tau
\end{array}
$$

Terms

$$
\begin{array}{lcl}
a & ::= & \\
 & | & x \\
 & | & \lambda(x : \tau)\ a \\
 & | & a\ a \\
 & | & \Lambda(\alpha \geqslant \tau)\ a \\
 & | & a\,\phi \\
 & | & \mathsf{let}\ x = a\ \mathsf{in}\ a
\end{array}
$$

Syntax

Instantiations

Types

$$
\begin{array}{lll}
\tau & ::= & \\
 & \mid & \alpha \\
 & \mid & \tau \to \tau \\
 & \mid & \forall\,(\alpha \geqslant \tau)\ \tau \\
 & \mid & \bot \\
\phi & ::= & \tau \\
 & \mid & !\alpha \\
 & \mid & \forall\,(\geqslant \phi) \\
 & \mid & \forall\,(\alpha \geqslant)\ \phi \\
 & \mid & \& \\
 & \mid & \parr \\
 & \mid & \phi; \phi \\
 & \mid & \mathbb{1}
\end{array}
$$

Terms

$$
\begin{array}{lll}
a & ::= & \\
 & \mid & x \\
 & \mid & \lambda(x : \tau)\ a \\
 & \mid & a\ a \\
 & \mid & \Lambda(\alpha \geqslant \tau)\ a \\
 & \mid & a\,\phi \\
 & \mid & \mathsf{let}\ x = a\ \mathsf{in}\ a
\end{array}
$$

Instantiation as a relation

UNDER

$$\frac{\Gamma, \alpha \geqslant \tau \vdash \phi : \tau_1 \leq \tau_2}{\Gamma \vdash \forall(\alpha \geqslant)\, \phi : \forall(\alpha \geqslant \tau)\, \tau_1 \leq \forall(\alpha \geqslant \tau)\, \tau_2}$$

BOT

$$\frac{}{\Gamma \vdash \tau : \bot \leq \tau}$$

INSIDE

$$\frac{\Gamma \vdash \phi : \tau_1 \leq \tau_2}{\Gamma \vdash \forall(\geqslant \phi) : \forall(\alpha \geqslant \tau_1)\, \tau \leq \forall(\alpha \geqslant \tau_2)\, \tau}$$

ABSTR

$$\frac{\alpha \geqslant \tau \in \Gamma}{\Gamma \vdash !\alpha : \tau \leq \alpha}$$

ELIM

$$\frac{}{\Gamma \vdash \& : \forall(\alpha \geqslant \tau)\, \tau' \leq \tau'\{\alpha \leftarrow \tau\}}$$

ID

$$\frac{}{\Gamma \vdash \mathbb{1} : \tau \leq \tau}$$

INTRO

$$\frac{\alpha \notin \mathsf{ftv}(\tau)}{\Gamma \vdash \wp : \tau \leq \forall(\alpha \geqslant \bot)\, \tau}$$

COMP

$$\frac{\Gamma \vdash \phi_1 : \tau_1 \leq \tau_2 \qquad \Gamma \vdash \phi_2 : \tau_2 \leq \tau_3}{\Gamma \vdash \phi_1; \phi_2 : \tau_1 \leq \tau_3}$$

Instantiation as a function

If $\Gamma \vdash \phi : \tau \leq \tau'$ then $\tau\ \phi = \tau'$

$$
\begin{array}{rlll}
\tau & (!\alpha) & = \alpha & \\
\bot & (\tau) & = \tau & \\
\tau & (\mathbb{1}) & = \tau & \\
\tau & (\aleph) & = \forall(\alpha \geqslant \bot)\,\tau & \alpha \notin \mathsf{ftv}(\tau) \\
\tau & (\phi_1;\phi_2) & = (\tau\ (\phi_1))\ (\phi_2) & \\
(\forall(\alpha \geqslant \tau)\,\tau') & (\&) & = \tau'\{\alpha \leftarrow \tau\} & \\
(\forall(\alpha \geqslant \tau)\,\tau') & (\forall(\geqslant \phi)) & = \forall(\alpha \geqslant \tau\ (\phi))\,\tau' & \\
(\forall(\alpha \geqslant \tau)\,\tau') & (\forall(\alpha \geqslant)\,\phi) & = \forall(\alpha \geqslant \tau)\,(\tau'\ (\phi)) &
\end{array}
$$

Instantiation as a function

If $\Gamma \vdash \phi : \tau \leq \tau'$ then $\tau\ \phi = \tau'$

$$
\begin{array}{rlll}
\tau & (!\alpha) & = & \alpha \\
\bot & (\tau) & = & \tau \\
\tau & (\mathbb{1}) & = & \tau \\
\tau & (\otimes) & = & \forall\,(\alpha \geqslant \bot)\ \tau \\
\tau & (\phi_1;\phi_2) & = & (\tau\ (\phi_1))\ (\phi_2) \\
(\forall\,(\alpha \geqslant \tau)\ \tau') & (\&) & = & \tau'\{\alpha \leftarrow \tau\} \\
(\forall\,(\alpha \geqslant \tau)\ \tau') & (\forall\,(\geqslant \phi)) & = & \forall\,(\alpha \geqslant \tau\ (\phi))\ \tau' \\
(\forall\,(\alpha \geqslant \tau)\ \tau') & (\forall\,(\alpha \geqslant)\ \phi) & = & \forall\,(\alpha \geqslant \tau)\ (\tau'\ (\phi))
\end{array}
$$

omitting $\alpha \geqslant \tau \in \Gamma$

$\alpha \notin \mathsf{ftv}(\tau)$

The functional view is complete, but sound only for well-typed instantiations.

Typing rules

$$\textsc{Var}\quad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \textsc{Abs}\quad \frac{\Gamma, x : \tau \vdash a : \tau'}{\Gamma \vdash \lambda(x : \tau)\, a : \tau \to \tau'} \qquad \textsc{App}\quad \frac{\Gamma \vdash a_1 : \tau_2 \to \tau_1 \qquad \Gamma \vdash a_2 : \tau_2}{\Gamma \vdash a_1\ a_2 : \tau_1}$$

$$\textsc{TAbs}\quad \frac{\Gamma, \alpha \geqslant \tau' \vdash a : \tau \qquad \alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \Lambda(\alpha \geqslant \tau')\, a : \forall(\alpha \geqslant \tau')\, \tau}$$

$$\textsc{TApp}\quad \frac{\Gamma \vdash a : \tau \qquad \Gamma \vdash \phi : \tau \leq \tau'}{\Gamma \vdash a(\phi) : \tau'}$$

Examples

Assume:

$$\begin{array}{rcl} \tau_{\mathsf{min}} & \triangleq & \forall\,(\alpha \geqslant \bot)\ \alpha \to \alpha \to \alpha \\ \tau_{\mathsf{cp}} & \triangleq & \forall\,(\alpha \geqslant \bot)\ \alpha \to \alpha \to \mathsf{bool} \\ \tau_{\mathsf{and}} & \triangleq & \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool} \\ \phi & \triangleq & \forall\,(\geqslant \mathsf{bool});\,\& \\ & = & \langle \mathsf{bool} \rangle \end{array} \qquad \text{with sugar } \langle \tau \rangle \triangleq (\forall\,(\geqslant \tau);\,\&)$$

Then:

$$\begin{array}{rcl} \phi & : & \tau_{\mathsf{min}} \leq \tau_{\mathsf{and}} \\ \phi & : & \tau_{\mathsf{cp}} \leq \tau_{\mathsf{and}} \end{array}$$

Assume:

$$\begin{array}{rcl} \tau_K & \triangleq & \forall\,(\alpha \geqslant \bot)\ \forall\,(\beta \geqslant \bot)\ \alpha \to \beta \to \alpha \\ \phi' & \triangleq & \forall\,(\alpha \geqslant)\ (\forall\,(\geqslant \alpha);\,\&) \\ & = & \forall\,(\alpha \geqslant)\ \langle \alpha \rangle \end{array}$$

Then:

$$\phi' \ : \ \tau_K \leq \tau_{\mathsf{min}}$$

Back to choice

Verify

$$\begin{array}{rcl} \mathit{id} & \triangleq & \Lambda(\alpha \geqslant \bot)\, \lambda(x : \alpha)\, x \\ & : & \forall\,(\alpha \geqslant \bot)\, \alpha \to \alpha \quad \triangleq \quad \tau_{\mathit{id}} \\ \mathsf{choice} & \triangleq & \Lambda(\beta \geqslant \bot)\, \lambda(x : \beta)\, \lambda(y : \beta)\, x \\ & : & \forall\,(\beta \geqslant \bot)\, \beta \to \beta \to \beta \\ \mathsf{choice\_id} & \triangleq & \Lambda(\beta \geqslant \tau_{\mathit{id}})\, \mathsf{choice}\, \langle\beta\rangle\, (\mathsf{id}\, (!\beta)) \\ & : & \forall\,(\beta \geqslant \tau_{\mathit{id}})\, \beta \to \beta \end{array}$$

Specializations

$$\mathsf{choice\_id}\, \& \quad : \quad (\forall\,(\alpha \geqslant \bot)\, \alpha \to \alpha) \to (\forall\,(\alpha \geqslant \bot)\, \alpha \to \alpha)$$

$$\begin{array}{rcl} \mathsf{choice\_id}\, (\otimes; \forall\,(\alpha \geqslant)\, (\forall\,(\geqslant \langle\alpha\rangle); \&)) & & \\ & : & \forall\,(\alpha \geqslant \bot)\, (\alpha \to \alpha) \to \alpha \to \alpha \\ & \longrightarrow & \Lambda(\alpha)\, \mathsf{choice\_id}\, (\forall\,(\geqslant \langle\alpha\rangle); \&) \end{array}$$

Reduction contexts

$$
\begin{array}{rcl}
E & ::= & [\cdot] \mid E\,\phi \mid \lambda(x:\tau)\,E \mid \Lambda(\alpha \geqslant \tau)\,E \\
& \mid & E\;a \mid a\;E \mid \mathsf{let}\;x = E\;\mathsf{in}\;a \mid \mathsf{let}\;x = a\;\mathsf{in}\;E
\end{array}
$$

Term reduction

$$
\begin{array}{lcll}
(\lambda(x:\tau)\ a_1)\ a_2 & \longrightarrow & a_1\{x \leftarrow a_2\} & \\
\mathsf{let}\ x = a_2\ \mathsf{in}\ a_1 & \longrightarrow & a_1\{x \leftarrow a_2\} & \\
\\
a\,\mathbb{1} & \longrightarrow & a & \\
a\,(\phi;\phi') & \longrightarrow & a\,\phi\,(\phi') & \\
a\,\wp & \longrightarrow & \Lambda(\alpha \geqslant \bot)\ a & \alpha \notin \mathsf{ftv}(\tau) \\
\\
(\Lambda(\alpha \geqslant \tau)\ a)\,\& & \longrightarrow & a\{!\alpha \leftarrow \mathbb{1}\}\{\alpha \leftarrow \tau\} & \\
(\Lambda(\alpha \geqslant \tau)\ a)\,(\forall\,(\alpha \geqslant)\ \phi) & \longrightarrow & \Lambda(\alpha \geqslant \tau)\ (a\,\phi) & \\
(\Lambda(\alpha \geqslant \tau)\ a)\,(\forall\,(\geqslant \phi)) & \longrightarrow & \Lambda(\alpha \geqslant \tau\,\phi)\ a\{!\alpha \leftarrow \phi; !\alpha\} &
\end{array}
$$

$$E[a] \longrightarrow E[a'] \qquad \text{if} \qquad a \longrightarrow a'$$

Contains System F as a subsystem

$$
\begin{aligned}
(\Lambda(\alpha)\ a)\ \langle\tau\rangle \quad &= \quad (\Lambda(\alpha \geqslant \bot)\ a)\ (\forall(\geqslant \tau); \&) \\
&\longrightarrow (\Lambda(\alpha \geqslant \bot)\ a)\ (\forall(\geqslant \tau))\ (\&) \\
&\longrightarrow (\Lambda(\alpha \geqslant \bot(\tau))\ a\{!\alpha \leftarrow \tau; !\alpha\})\ (\&) \\
&= \quad (\Lambda(\alpha \geqslant \tau)\ a)\ (\&) \\
&\longrightarrow a\{\alpha \leftarrow \tau\}
\end{aligned}
$$

Reduction of choice_id

skip back

$$(\Lambda(\beta \geqslant \sigma)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta))$$
$$(\otimes; \forall\,(\gamma \geqslant)\ (\forall\,(\geqslant\langle\gamma\rangle); \&))$$

$$a\,(\phi; \phi') \longrightarrow a\,\phi\,(\phi') \qquad (\iota\text{-}\textsc{Seq})$$

Reduction of choice_id

skip back

$$(\Lambda(\beta \geqslant \sigma)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta))$$
$$(\otimes)\ \ (\forall\,(\gamma \geqslant)\ (\forall\,(\geqslant\langle\gamma\rangle);\&))$$

$$a\otimes \longrightarrow \Lambda(\alpha \geqslant \bot)\ a \qquad \alpha \notin \mathsf{ftv}(\tau) \qquad (\iota\text{-}\textsc{Intro})$$

Reduction of choice_id

skip back

$$(\Lambda(\alpha \geqslant \bot)\ \Lambda(\beta \geqslant \sigma)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta))$$
$$(\forall\,(\gamma \geqslant)\ (\forall\,(\geqslant \langle\gamma\rangle); \&))$$

$$(\Lambda(\alpha \geqslant \tau)\ a)\ (\forall\,(\alpha \geqslant)\ \phi) \longrightarrow \Lambda(\alpha \geqslant \tau)\ (a\,\phi) \quad (\iota\text{-}\textsc{Under})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)\ \Big(\Lambda(\beta \geqslant \sigma)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta)\Big)\ (\forall\,(\geqslant \langle\alpha\rangle); \&)$$

$$a\,(\phi; \phi') \longrightarrow a\,\phi\,(\phi') \qquad (\iota\text{-}\textsc{Seq})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)\ \Big(\Lambda(\beta \geqslant \sigma)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta)\Big)\ (\forall\,(\geqslant \langle\alpha\rangle))\ (\&)$$

$$(\Lambda(\alpha \geqslant \tau)\ a)\,(\forall\,(\geqslant \phi)) \longrightarrow \Lambda(\alpha \geqslant \tau\,\phi)\ a\{!\alpha \leftarrow \phi; !\alpha\}$$

(ι-Inside)

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$\left(\Lambda(\beta \geqslant \sigma\ \langle\alpha\rangle)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (!\beta)\right)$$
$$\{!\beta \leftarrow\ \langle\alpha\rangle; !\beta\}\ (\&)$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)\left(\Lambda(\beta \geqslant \alpha \to \alpha)\ \mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\ (\langle\alpha\rangle; !\beta)\right)\ (\&)$$

$$(\Lambda(\alpha \geqslant \tau)\ a)\,\& \longrightarrow a\{!\alpha \leftarrow \mathbb{1}\}\{\alpha \leftarrow \tau\} \qquad (\iota\text{-Elim})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)\left(\mathsf{choice}\ \langle\beta\rangle\ \mathsf{id}\,(\langle\alpha\rangle; !\beta)\right)\{!\beta \leftarrow \mathbb{1}\}\{\beta \leftarrow \alpha \to \alpha\}$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$\quad \mathsf{choice}\ \langle \alpha \to \alpha \rangle\ \mathsf{id}\ (\langle \alpha \rangle; \mathbb{1})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$(\Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \text{if } \mathit{true} \text{ then } x \text{ else } y)$$
$$\langle \alpha \to \alpha \rangle$$
$$(\Lambda(\alpha)\ \lambda(x : \alpha)\ x)\ (\langle \alpha \rangle; \mathbb{1})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$(\Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \text{if } \mathit{true} \text{ then } x \text{ else } y)$$
$$\langle \alpha \to \alpha \rangle$$
$$(\lambda(x : \alpha)\ x)$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$(\Lambda(\beta)\ \lambda(x : \beta)\ \lambda(y : \beta)\ \text{if } \mathit{true} \text{ then } x \text{ else } y)$$
$$\langle \alpha \to \alpha \rangle$$
$$(\Lambda(\alpha)\ \lambda(x : \alpha)\ x)\ \langle \alpha \rangle\ (\mathbb{1})$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$(\lambda(x : \alpha \to \alpha)\ \lambda(y : \alpha \to \alpha)$$
$$\text{if } \mathit{true} \text{ then } x \text{ else } y)$$
$$(\lambda(x : \alpha)\ x)$$

Reduction of choice_id

skip back

$$\Lambda(\alpha \geqslant \bot)$$
$$\lambda(y : \alpha \to \alpha) \text{ if } \mathit{true} \text{ then } \lambda(x : \alpha)\ x \text{ else } y$$

Properties of reduction

Reduction in any context

- It is **confluent** *on well-typed terms*
  (Reduction of ill-typed terms may block on type errors while some other reduction path may jump over the type error)
- It *should be* terminating (but we have not proved it).
- It is **type preserving**.

Call-by-value and call-by-name reduction strategies

- Subject reduction holds,
- Progress holds.

Elaboration of $e\text{ML}^\text{F}$ into $x\text{ML}^\text{F}$

Type preserving and semantics preserving translation

- This ensures type soundness of $e\text{ML}^\text{F}$.
- No computational overhead
  (by comparisson with translating $e\text{ML}^\text{F}$ into System F)

The translation

- Can be computed during inference by instrumenting the constraint solver.
- I will save you the details...

Elaboration of $e\mathrm{ML}^\mathsf{F}$ into $x\mathrm{ML}^\mathsf{F}$

Elaboration is not surjective

- $x\mathrm{ML}^\mathsf{F}$ is strictly more expressive than $e\mathrm{ML}^\mathsf{F}$!

  (there are terms of $x\mathrm{ML}^\mathsf{F}$ that do not have any counterpart in $e\mathrm{ML}^\mathsf{F}$)

$$\begin{array}{lcl} \tau_0 & \triangleq & \forall\,(\alpha \geqslant \tau)\,\forall\,(\beta \geqslant \alpha)\,\beta \to \alpha \\ \tau_{\mathsf{id}} & \triangleq & \forall\,(\alpha \geqslant \tau)\,\alpha \to \alpha \end{array}$$

▸ See more?

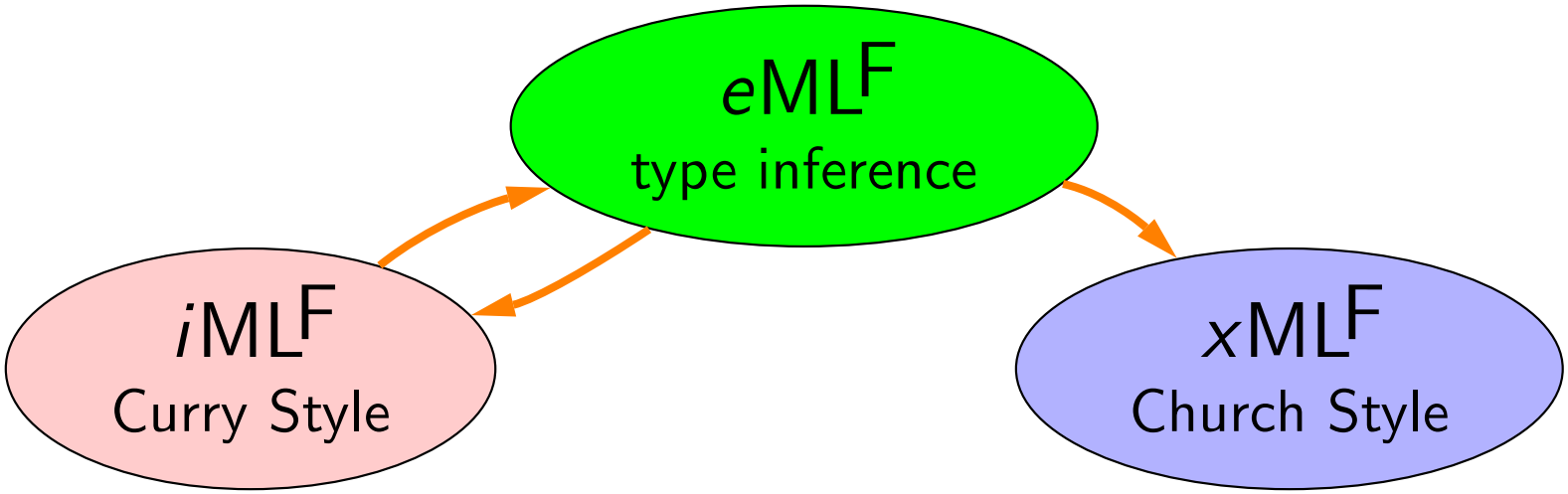
In $e\mathrm{ML}^\mathsf{F}$ $\tau_0 = \tau_{\mathsf{id}}$
In $x\mathrm{ML}^\mathsf{F}$ $\tau_0 \leq \tau_{\mathsf{id}}$ (strictly)

Good choice for type inference
Natural choice otherwise.

- The subset of $x\mathrm{ML}^\mathsf{F}$ that is the target of $e\mathrm{ML}^\mathsf{F}$ is stable by reduction.
  (But its specification, via a restriction of typing of instantiations, is not very natural)

Conclusion

The ML^F trilogy

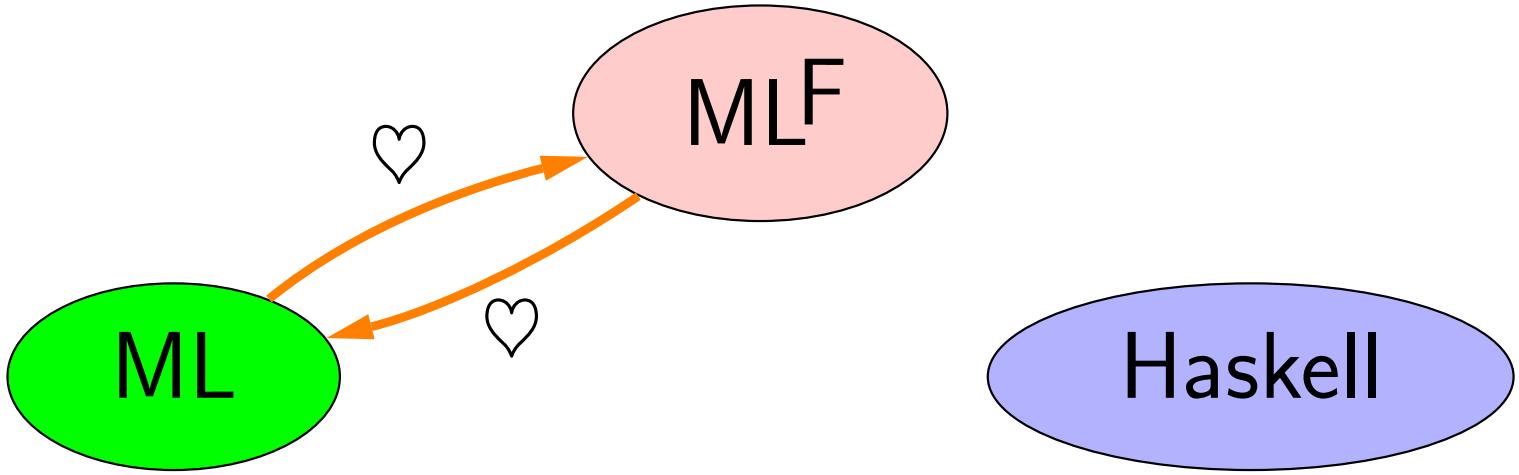


Beyond ML^F

Is there a language of coercions that is an extension of both $x\mathrm{ML}^\mathrm{F}$ and F^η?

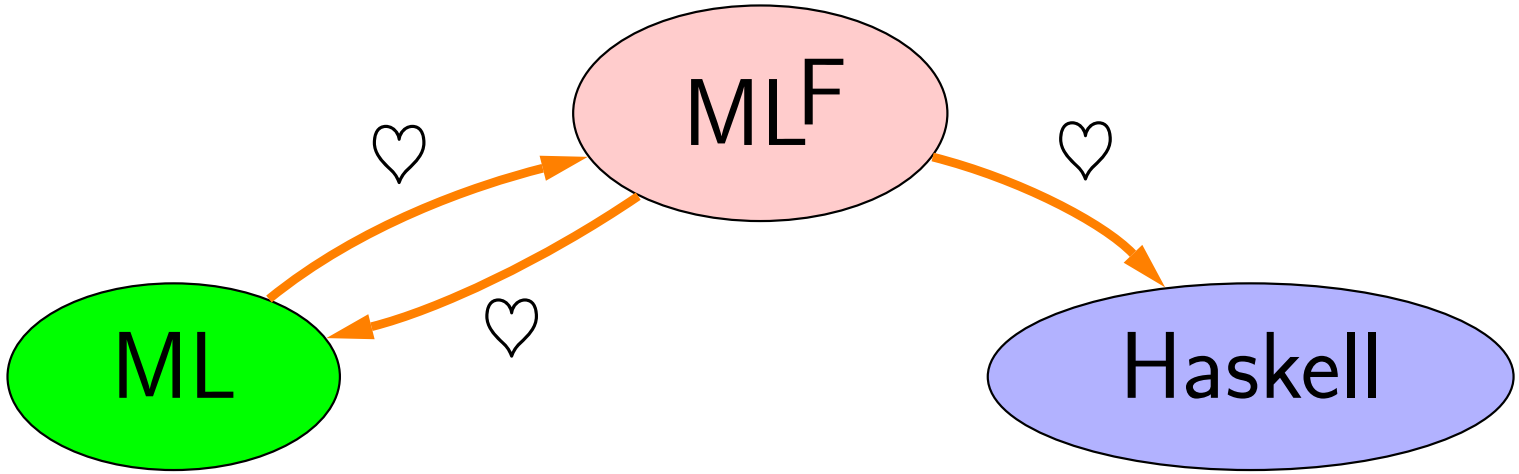
Epilogue

The ML^F drama



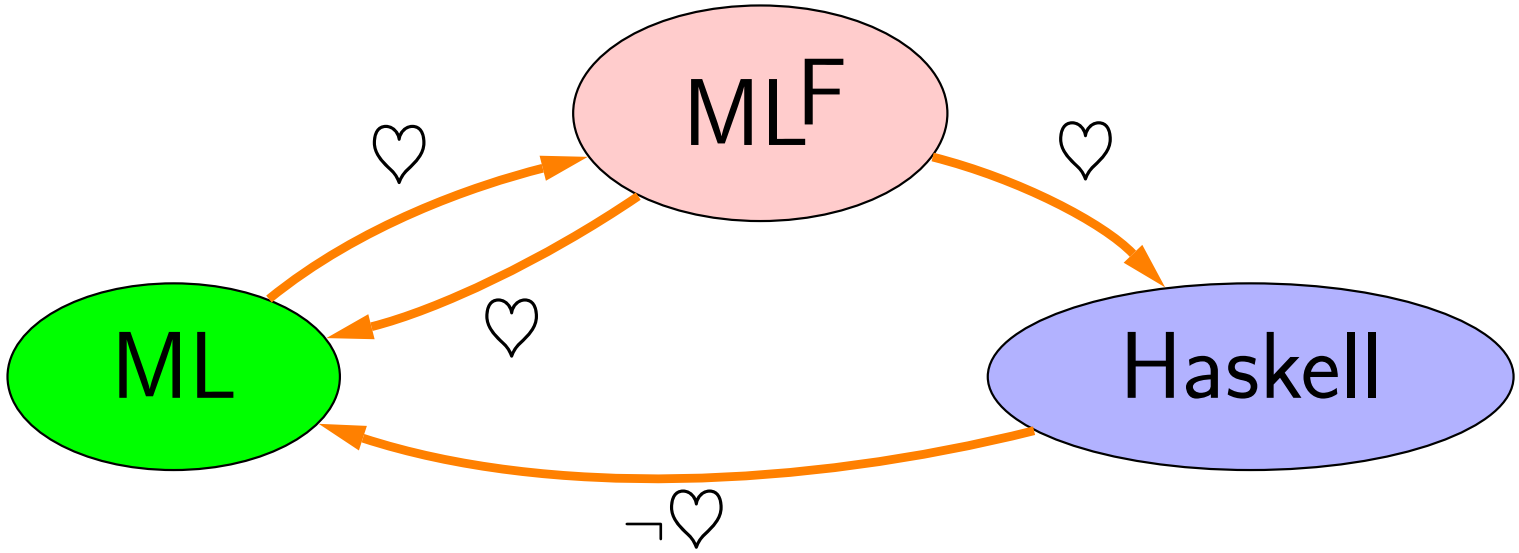
Epilogue

The ML^F drama



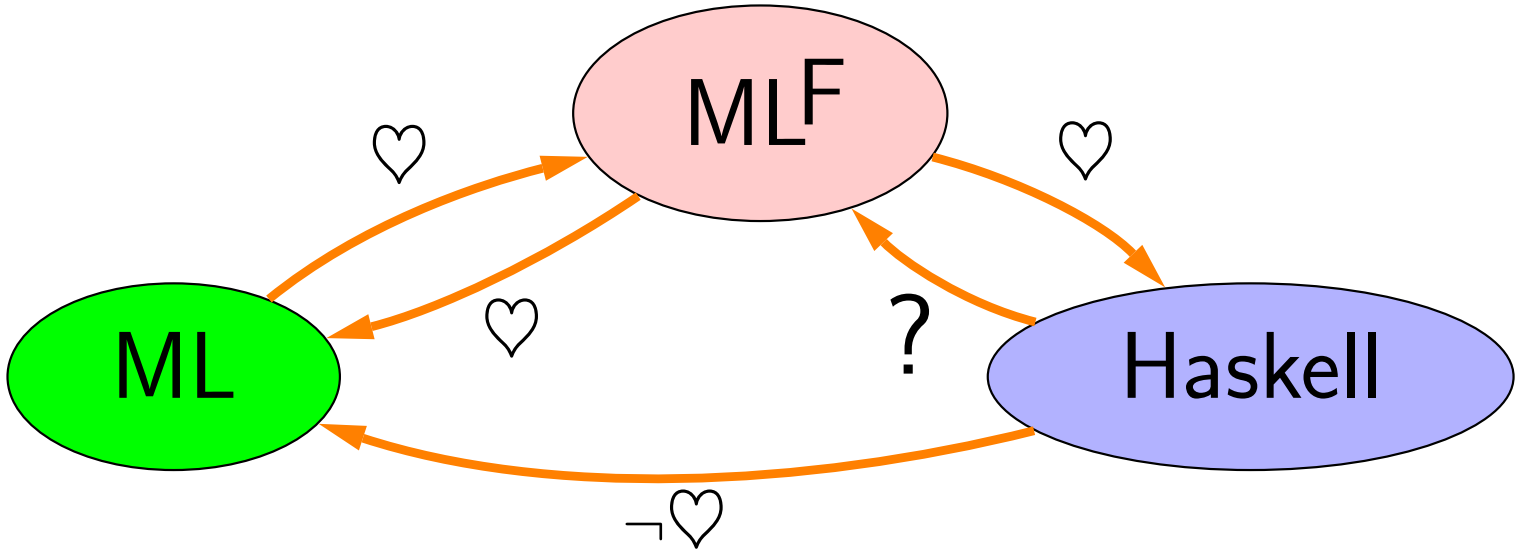
Epilogue

The ML^F drama



Epilogue

The ML^F drama



Appendix

F^η : system F with retyping functions

Definition F^η is the closure of F by η-conversion.

Typechecking is as in F with the type containment rule

$$\frac{\Gamma \vdash a : \tau \qquad \tau \triangleright \tau'}{\Gamma \vdash a : \tau}$$

Type containment $\triangleright$ is the smallest reflexive, transitive relation that is

- covariant on all constructors except on the left of arrows where it is contravariant,
- $\forall\,(\overline{\alpha})\ \tau \triangleright \forall\,(\overline{\alpha}')\ \tau\{\overline{\alpha} \leftarrow \overline{\tau}'\}$ whenever $\overline{\alpha}' \notin \mathrm{ftv}(\forall\,(\overline{\alpha})\ \tau)$,
- $\forall$ distributes over $\rightarrow$, *i.e.* $\forall\,(\alpha)\ (\tau \rightarrow \tau') \triangleright \forall\,(\alpha)\ \tau \rightarrow \forall\,(\alpha)\ \tau'$.

Property

$\tau \triangleright \tau'$ if and only if there exists $a \in F$ of type $\tau \rightarrow \tau'$ that $\beta\eta$-reduces to the identity; s is called a retyping function.

Good

F^η allows *implicit* coercions of type expressions, according to variances:

- type-expressions in covariant possitions can be instantiated.
- type-expressions in contracovariant possitions can be generalized.

F^η has *more-principal* types and more *principal-types* than System-F.
For instance, in F^η we have

$$\forall(\alpha)\,((\alpha \to \alpha) \to (\alpha \to \alpha)) \leq (\forall(\alpha)\,\alpha \to \alpha) \to (\forall(\alpha)\,\alpha \to \alpha)$$

Still

F^η does not allow simultaneous instantiations of subterms.
In F^η, there is no counterpart to the ML^F type:

$$\forall(\beta \geqslant \tau_{id})\,\beta \to \beta \to \beta$$

eML^F and F^η instance relations are incomparable

$x\mathsf{ML}^\mathsf{F} \not\subseteq e\mathsf{ML}^\mathsf{F}$

Intuition

◀ back

In $i\mathsf{ML}^\mathsf{F}$:

$$\begin{array}{rcl} \tau_0 & \triangleq & \forall\,(\alpha \geqslant \tau)\,\forall\,(\beta \geqslant \alpha)\,\beta \to \alpha \\ \tau_{\mathsf{id}} & \triangleq & \forall\,(\alpha \geqslant \tau)\,\alpha \to \alpha \\ [\![\tau_0]\!] = [\![\tau_{\mathsf{id}}]\!] & = & \{\tau' \to \tau' \mid \tau \leq \tau'\} \\ [\![\forall\,(\beta \geqslant \tau)\,\beta \to \tau]\!] & = & \{\tau' \to \tau \mid \tau \leq \tau'\} \end{array}$$

In $x\mathsf{ML}^\mathsf{F}$:

$$[\![\tau_0]\!] \;=\; \{\tau'' \to \tau' \mid \tau \leq \tau' \leq \tau''\}$$

In graphical notation τ_0 cannot be written—aliases must be inlined (substituted).

$x\mathsf{ML}^\mathsf{F} \not\subseteq e\mathsf{ML}^\mathsf{F}$

Example

◀ back

In $x\mathsf{ML}^\mathsf{F}$ choice may be given types

$$\begin{array}{rcl} \tau'_{\mathsf{ch}} & \triangleq & \forall(\alpha)\,\forall(\alpha' \geqslant \alpha)\,\forall(\alpha'' \geqslant \alpha')\;\alpha \to \alpha' \to \alpha'' \\ \tau''_{\mathsf{ch}} & \triangleq & \forall(\alpha)\,\forall(\alpha' \geqslant \alpha)\,\forall(\alpha'' \geqslant \alpha)\;\alpha' \to \alpha \to \alpha'' \end{array}$$

which are incomparable and both more general than:

$$\tau_{\mathsf{ch}} \quad \triangleq \quad \forall(\alpha)\;\alpha \to \alpha \to \alpha$$

Typing choice with τ_{ch} or τ'_{ch} does not allow to apply choice to more arguments. However, it allows to pass choice to a function that expects an argument of both type $\tau_1 \to \tau'_1 \to \tau''_1$ and $\tau'_2 \to \tau'_2 \to \tau''_2$ with $\tau_i \leq \tau'_i \leq \tau''_i$ for i in $\{1, 2\}$.

Here, the type τ'_{ch} may be recovered from the type τ_{ch} by η-expansion, as:

$$\begin{array}{l} \Lambda(\alpha)\,\Lambda(\beta \geqslant \alpha)\,\Lambda(\gamma \geqslant \beta) \\ \qquad \lambda(x : \alpha)\,\lambda(y : \beta)\,\mathsf{choice}\,\gamma\,(x\,(!\beta; !\gamma))\,(y\,(!\gamma)) \end{array}$$