Contents

A framework for Partial type inference in the Predicative Fragment of System F^η

Didier Rémy

INRIA-Rocquencourt

Joint work with François Pottier

Sophia, Mars 2004

Reminder: ML is great

ML

- Simply-typed λ-calculus with let-bindings

Simple type inference

- *Intuitively*, similar to simply-typed λ-calculus (relies on first-order unification) plus generalization of inferred types at let-nodes.
- *In fact*, some technicalites, proofs often ommitted...

Yet quite expressive

- Outermost quantification already buys you a lot.

ML is a local optimum

Observation

- Twenty years later, ML is still great, but its limitations appear more problematic...

  Many extensions calls for first-class polymorphism: objects, monads, GADT's... modules? (In fact, just think of existential types)

  The extensions are often twisted to fit within the ML box.

- System F (second-order types) is quite powerful and still simple, but it misses type inference.

- Solution: Keep type inference... as far as possible...
  but have second-order types!

ML is aging...

Observation

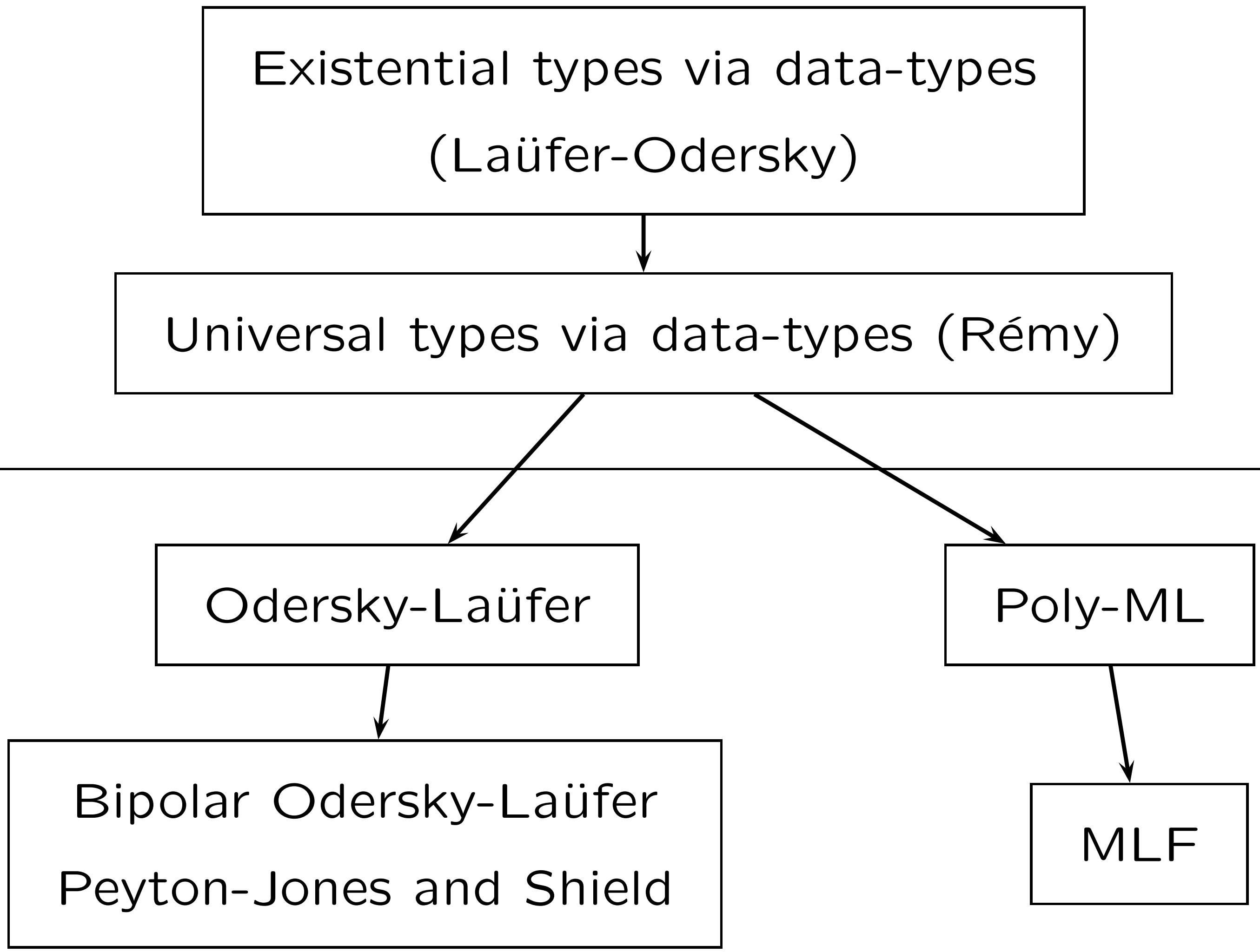
- Solution: Keep type inference... as far as possible...
  but have second-order types!

Two approaches for partial type inference

- From System F towards ML
  - Use second-order unification [Frank Pfenning].
    Expressive, but undecidable. Places for type abtraction and type application must still be explicit.
  - Use Local type inference [Pierce Turner, Odersky-Zenger-Zenger] to remove the most dummy type annotations. Not conservative over ML.
- From ML towards System F...

Extending ML with higher-order types

ML



(almost)
transparent
boxed

Predicative... *vs* Impredicative...

Polymorphism

Typing rules for System F^X

Var

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

App

$$\frac{\Gamma \vdash a_1 : \sigma_2 \rightarrow \sigma_1 \qquad \Gamma \vdash a_2 : \sigma_2}{\Gamma \vdash a_1\ a_2 : \sigma_1}$$

Fun

$$\frac{\Gamma, z : \sigma \vdash a : \sigma'}{\Gamma \vdash \mathsf{fun}\ (z)\ a : \sigma \rightarrow \sigma'}$$

Gen

$$\frac{\Gamma \vdash a : \sigma \qquad \alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash a : \forall\, \alpha.\, \sigma}$$

Inst

$$\frac{\Gamma \vdash a : \sigma \qquad \sigma \leq_X \sigma'}{\Gamma \vdash a : \sigma'}$$

- Amazingly simple specification!
- Parameterized by an instance relation $\leq_X$ called type containment.

Terms

$$\mathsf{t} ::= \mathsf{x} \mid \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} \mid \mathsf{t}_1\ \mathsf{t}_2$$

$$\mathsf{x} ::= \mathsf{z} \mid \mathsf{c}$$

Types

$$\sigma ::= \alpha \mid \sigma \to \sigma \mid \forall\, \alpha .\, \sigma$$

Type-containment $\leq$

$\leq$ for System F

The smallest relation $\leqslant$ that satisfies the rules:

Sub

$$\frac{\bar\beta \notin \mathsf{ftv}(\forall\,\bar\alpha.\,\sigma)}{\forall\,\bar\alpha.\,\sigma \leqslant \forall\,\bar\beta.\,\sigma[\bar\sigma/\bar\alpha]}$$

Type-containment $\leq^\eta$ (Mitchell 1984)

$\leq^\eta$ for System F^η $\quad = \quad$ System F modulo η-expansion.

The smallest relation $\leqslant$ that satisfies the rules:

Sub

$$\frac{\bar\beta \notin \mathsf{ftv}(\forall\,\bar\alpha.\,\sigma)}{\forall\,\bar\alpha.\,\sigma \leqslant \forall\,\bar\beta.\,\sigma[\bar\sigma/\bar\alpha]}$$

Trans

$$\frac{\sigma \leqslant \sigma' \qquad \sigma' \leqslant \sigma''}{\sigma \leqslant \sigma''}$$

Arrow

$$\frac{\sigma_1' \leqslant \sigma_1 \qquad \sigma_2 \leqslant \sigma_2'}{\sigma_1 \rightarrow \sigma_2 \leqslant \sigma_1' \rightarrow \sigma_2'}$$

All

$$\frac{\sigma \leqslant \sigma'}{\forall\,\alpha.\,\sigma \leqslant \forall\,\alpha.\,\sigma'}$$

Distrib

$$\forall\,\alpha.\,\sigma \rightarrow \sigma' \leqslant (\forall\,\alpha.\,\sigma) \rightarrow \forall\,\alpha.\,\sigma'$$

Type-containment $\leq^\eta$ (Mitchell 1984)

$\leq^\eta$ for System F$^\eta$ = System F modulo η-expansion.

The smallest relation $\leqslant$ that satisfies the rules:

Sub

$$\frac{\bar\beta \notin \mathsf{ftv}(\forall\,\bar\alpha.\,\sigma)}{\forall\,\bar\alpha.\,\sigma \leqslant \forall\,\bar\beta.\,\sigma[\bar\sigma/\bar\alpha]}$$

Trans

$$\frac{\sigma \leqslant \sigma' \qquad \sigma' \leqslant \sigma''}{\sigma \leqslant \sigma''}$$

Arrow

$$\frac{\sigma_1' \leqslant \sigma_1 \qquad \sigma_2 \leqslant \sigma_2'}{\sigma_1 \to \sigma_2 \leqslant \sigma_1' \to \sigma_2'}$$

All

$$\frac{\sigma \leqslant \sigma'}{\forall\,\alpha.\,\sigma \leqslant \forall\,\alpha.\,\sigma'}$$

Distrib

$$\forall\,\alpha.\,\sigma \to \sigma' \leqslant (\forall\,\alpha.\,\sigma) \to \forall\,\alpha.\,\sigma'$$

$\alpha \notin \mathsf{ftv}(\sigma')$

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

Distrib-Left

$$\forall\,\alpha.\,\sigma \to \sigma' \leqslant (\forall\,\alpha.\,\sigma) \to \sigma'$$

Distrib-Right

$$\forall\,\alpha.\,\sigma \to \sigma' \leqslant \sigma \to \forall\,\alpha.\,\sigma'$$

(also $\geqslant$, hence $\equiv$, by Sub+All)

Properties

Type Soundness

Both F and F^{η} have subject reduction
(+ progress if constants are added)

Type Inference

- Neither allows type inference
- $\leq^{\eta}$ itself is not decidable.

F^{η} is better suited for type inference (suggestion by Mitchell)

Value restriction

Well-known in ML: restrict generalization to syntactic values
(or non-expansive expressions)

$$\text{Gen}_v \quad \frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \Gamma \qquad \mathsf{t} \in \mathcal{U}}{\Gamma \vdash \mathsf{t} : \forall \bar{\alpha}.\, \sigma}$$

(For instance, model references with a global store)

What about side effects?

Value restriction

Well-known in ML: restrict generalization to syntactic values
(or non-expansive expressions)

Gen$_v$

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \Gamma \qquad \mathsf{t} \in \mathcal{U}}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

Not sufficient, because

$$\forall\, \alpha.\, \sigma' \to \sigma \leq^{\eta} \sigma' \to \forall\, \alpha.\, \sigma \qquad\qquad \bar{\alpha} \notin \mathsf{ftv}(\sigma)$$

not valid with side effects. Must be removed.

Enhanced Value restriction (Garrigue)

Well-known in ML: restrict generalization to syntactic values
(or non-expansive expressions)

Gen$_v$

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \Gamma \qquad \mathsf{t} \in \mathcal{U}}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

Not sufficient, because

$$\forall\, \alpha.\, \sigma' \to \sigma \leq^{\eta} \sigma' \to \forall\, \alpha.\, \sigma \qquad\qquad \bar{\alpha} \notin \mathsf{ftv}(\sigma)$$

not valid with side effects. Must be removed.

Two weak

(fun (z) [ ]) () : list α with α non genralizable.
But is is *safe* and *useful* to generalize α!

Value restriction

Well-known in ML: restrict generalization to syntactic values
(or non-expansive expressions) or unipolar type variables

Gen$_v$

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar\alpha \notin \Gamma \qquad \mathsf{t} \in \mathcal{U} \;\vee\; \bar\alpha \in \mathsf{ftv}^{\pm}(\sigma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar\alpha.\, \sigma}$$

$$\begin{cases} \mathsf{ftv}^{+}(\sigma) \setminus \mathsf{ftv}^{-}(\sigma) \\ \mathsf{ftv}^{-}(\sigma) \setminus \mathsf{ftv}^{+}(\sigma) \end{cases}$$

Not sufficient, because

$$\forall\, \alpha.\, \sigma' \to \sigma \leq^{\eta} \sigma' \to \forall\, \alpha.\, \sigma \qquad \bar\alpha \notin \mathsf{ftv}(\sigma), \bar\alpha \in \mathsf{ftv}^{\pm}(\sigma)$$

Two weak

(fun (z) [ ]) () : list α with α non genralizable.

But is is *safe* and *useful* to generalize α!

F^η_v with side effects and enhanced value restriction is sound.

Intuition

$$
\dfrac{
  \dfrac{
    \dfrac{\overset{\text{Substitution lemma}}{\Gamma, M \vdash \mathsf{v} : \mathsf{list}\ \alpha}}{\Gamma, \varphi(M) \vdash \mathsf{v} : \mathsf{list}\ (\forall\, \alpha.\, \alpha^{a})}
    \qquad
    \overset{\text{Covariance+Sub}}{\mathsf{list}\ (\forall\, \alpha.\, \alpha) \leq^{\eta} \mathsf{list}\ (\alpha)}
  }{\Gamma, \varphi(M) \vdash \mathsf{v} : \mathsf{list}\ (\alpha)}\ \text{Inst}
  \qquad
  \alpha \notin \Gamma, \varphi(M)
}{\Gamma, \varphi(M) \vdash \mathsf{v} : \forall\, \alpha.\, \mathsf{list}\ (\alpha)}\ \text{Gen}_v
$$

where $\varphi = \alpha \mapsto \forall\, \alpha.\, \alpha$

Generalizes Garrigue's result to F^η

[a] Positive occurrence

F_p: Predicative System F

A better candidate for type inference a la ML

Types in System F_p

$$\tau ::= \alpha \mid \tau \to \tau \qquad \text{monotypes}$$

$$\sigma ::= \tau \mid \sigma \to \sigma \mid \forall\, \alpha.\, \sigma \qquad \text{(poly)types}$$

Types variables may only be instantiated by monotypes

Rule Sub must be replaced by

$$\textsf{Sub}_p \quad \frac{\bar{\beta} \notin \mathsf{ftv}(\forall\, \bar{\alpha}.\, \sigma)}{\forall\, \bar{\alpha}.\, \sigma \leqslant \forall\, \bar{\beta}.\, \sigma[\bar{\tau}/\bar{\alpha}]}$$

F_p^η: Predicative System F^η

A better candidate for type inference a la ML

Types in System F_p

$$\tau ::= \alpha \mid \tau \to \tau \qquad \text{monotypes}$$

$$\sigma ::= \tau \mid \sigma \to \sigma \mid \forall\, \alpha.\, \sigma \qquad \text{(poly)types}$$

Types variables may only be instantiated by monotypes

Rule Sub must be replaced by

Sub$_p$

$$\frac{\bar\beta \notin \mathsf{ftv}(\forall\, \bar\alpha.\, \sigma)}{\forall\, \bar\alpha.\, \sigma \leqslant \forall\, \bar\beta.\, \sigma[\bar\tau/\bar\alpha]}$$

A significant restriction

- Encoding of existentials:

  When hiding $\sigma[\tau/\beta]$ as $\exists\,\beta.\,\sigma$, τ must be a monotype.

- Encoding of objects

  Objets are (at least) as complicated as

$$\exists\,\alpha_R.\,(\alpha_R \times (\alpha_R \to \sigma_M))$$

  where α_R hides the states. The state would have to be monomorphic, *i.e.* not contain objets.

A significant restriction

- apply (or map, iter, *etc.*)

$$\text{let apply} = \text{fun } (\text{f}) \text{ fun } (\text{z}) \text{ f z in } \ldots$$

will only take monomorphic arguments...

Need one version per polymorphic *shape* of the type of f ...
including one version per polymophic shape of the type of z,

which is really bad!

(all abstract types are incompatible)

A significant restriction

- apply (or map, iter, *etc.*)

$$\text{let apply} = \text{fun (f) fun (z) f z in } \ldots$$

will only take monomorphic arguments...

Need one version per polymorphic *shape* of the type of f . . .
including one version per polymophic shape of the type of z,

which is really bad!

(all abstract types are incompatible)

Can/must be combined with boxed polymorphism

Embed the impredicative fragment within data-types, as usual.

Not elegant. Still better than either one alone.

Expressions

$$t ::= x \mid \text{fun}\ (z)\ t \mid t_1 \quad t_2 \qquad \mid \text{let}\ z = \ t_1 \qquad \text{in}\ t_2$$

Expressions

$$t ::= x \mid \text{fun } (z)\ t \mid t_1\ (t_2 : \theta) \mid \text{let } z = (t_1 : \theta) \text{ in } t_2$$

Annotations

$$\theta ::= \exists\, \bar{\beta}.\, \sigma$$

Annotations are there to **explicitly specify the polymorphic shape of types** and let type inference **guess the monomorphic parts**.

Hence $\exists\, \bar{\beta}.$ plays as key a role (leaves room for guessing) as σ.

The monomorphic structure is always hanging off under some (possibly empty) polymorphic structure.

An empty annotation is $\exists\, \beta.\, \beta$

Typing rules for ML

Var

$$\frac{\mathsf{x} : \sigma \in \Gamma}{\Gamma \vdash \mathsf{x} : \sigma}$$

Inst

$$\frac{\Gamma \vdash \mathsf{t} : \sigma' \qquad \sigma' \leq \sigma}{\Gamma \vdash \mathsf{t} : \sigma}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar{\alpha}. \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \tau_2 \rightarrow \tau \qquad \Gamma \vdash \mathsf{t}_2 : \tau_2}{\Gamma \vdash \mathsf{t}_1\ \mathsf{t}_2 : \tau}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \tau_2 \vdash \mathsf{t} : \tau_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau_2 \rightarrow \tau_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall \bar{\alpha}. \tau_1 \qquad \Gamma, x : \forall \bar{\alpha}. \tau_1 \vdash \mathsf{t}_2 : \tau_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \tau_2}$$

Typing rules for ML with annotations

Var

$$\frac{\mathsf{x} : \sigma \in \Gamma}{\Gamma \vdash \mathsf{x} : \sigma}$$

Inst

$$\frac{\Gamma \vdash \mathsf{t} : \sigma' \qquad \sigma' \leq \sigma}{\Gamma \vdash \mathsf{t} : \sigma}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar\alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar\alpha. \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \tau_2[\bar\tau/\bar\beta] \rightarrow \tau \qquad \Gamma \vdash \mathsf{t}_2 : \tau_2[\bar\tau/\bar\beta]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists \bar\beta. \tau_2) : \tau}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \tau_2 \vdash \mathsf{t} : \tau_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau_2 \rightarrow \tau_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall \bar\alpha. \tau_1[\bar\tau/\bar\beta] \qquad \Gamma, x : \forall \bar\alpha. \tau_1[\bar\tau/\bar\beta] \vdash \mathsf{t}_2 : \tau_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists \bar\beta. \tau_1)\ \mathsf{in}\ \mathsf{t}_2 : \tau_2}$$

Typing rules for $\mathsf{F}_p^{\Downarrow}$

Var

$$\frac{\mathsf{x} : \sigma \in \Gamma}{\Gamma \vdash \mathsf{x} : \sigma}$$

Inst

$$\frac{\Gamma \vdash \mathsf{t} : \sigma' \qquad \sigma' \leq_p^{\shortparallel} \sigma}{\Gamma \vdash \mathsf{t} : \sigma}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar\alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar\alpha.\, \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar\tau/\bar\beta] \to \sigma \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar\tau/\bar\beta]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta.\, \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall\, \bar\alpha.\, \sigma_1[\bar\tau/\bar\beta] \qquad \Gamma, x : \forall\, \bar\alpha.\, \sigma_1[\bar\tau/\bar\beta] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Read $\Gamma \vdash \mathsf{t} : \sigma$ as checking rules: Γ, t, and σ given.

All types are of the form $\sigma[\bar\tau/\bar\beta]$ where σ is never guessed and $\bar{\bar\tau}$ is.

We get ML when all σ are β.

Var

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

Inst

$$\frac{\Gamma \vdash t : \sigma' \qquad \sigma' \leq_p^{\shortparallel} \sigma}{\Gamma \vdash t : \sigma}$$

Gen

$$\frac{\Gamma \vdash t : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash t : \forall \bar{\alpha}. \sigma}$$

App

$$\frac{\Gamma \vdash t_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \rightarrow \sigma \qquad \Gamma \vdash t_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash t_1\ (t_2 : \exists \bar{\beta}. \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, z : \sigma_2 \vdash t : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (z)\ t : \sigma_2 \rightarrow \sigma_1}$$

Let

$$\frac{\Gamma \vdash t_1 : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \vdash t_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ z = (t_1 : \exists \bar{\beta}. \sigma_1)\ \mathsf{in}\ t_2 : \sigma_2}$$

To prepare for type inference:

Put derivations in canonical, syntax-directed form.

Var

$$\frac{\mathsf{x} : \sigma \in \Gamma}{\Gamma \vdash \mathsf{x} : \sigma}$$

Inst

$$\frac{\Gamma \vdash \mathsf{t} : \sigma' \qquad \sigma' \leq_p^{\shortparallel} \sigma}{\Gamma \vdash \mathsf{t} : \sigma}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar{\alpha}. \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \sigma \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}. \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall\, \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall\, \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}. \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Merge the two rules.

$\mathsf{Inst}(\mathsf{R}(D)) \rightsquigarrow \mathsf{R}(\mathsf{Inst}(D))$, except when R is Var.

Typing rules for $\mathsf{F}_p^{\Downarrow}$

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \sigma}{\Gamma \vdash \mathsf{x} : \sigma}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \sigma \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall\, \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall\, \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Typing rules for $F_p^{\Downarrow}$

Var-Inst

$$\frac{x : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \sigma}{\Gamma \vdash x : \sigma}$$

Gen

$$\frac{\Gamma \vdash t : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash t : \forall \bar{\alpha}. \sigma}$$

App

$$\frac{\Gamma \vdash t_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \sigma \qquad \Gamma \vdash t_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash t_1\ (t_2 : \exists \bar{\beta}. \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, z : \sigma_2 \vdash t : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (z)\ t : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash t_1 : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \vdash t_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ z = (t_1 : \exists \bar{\beta}. \sigma_1)\ \mathsf{in}\ t_2 : \sigma_2}$$

Change σ into ρ.

A ρ is a σ without outer quantifiers.

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar{\alpha}. \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \sigma \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists \bar{\beta}. \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall \bar{\alpha}. \sigma_1[\bar{\tau}/\bar{\beta}] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists \bar{\beta}. \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \sigma \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \sigma}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall\, \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall\, \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Change σ_1 into ρ_1, since

$$\mathsf{App}(D_1, D_2) \rightsquigarrow \mathsf{Gen}(\mathsf{App}(\mathsf{Inst}(D_1), D_2))$$

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar\alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar\alpha.\, \sigma}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar\tau/\bar\beta] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar\tau/\bar\beta]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta.\, \sigma_2) : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash \mathsf{t}_1 : \forall\, \bar\alpha.\, \sigma_1[\bar\tau/\bar\beta] \qquad \Gamma, x : \forall\, \bar\alpha.\, \sigma_1[\bar\tau/\bar\beta] \vdash \mathsf{t}_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Var-Inst

$$\frac{x : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash x : \rho}$$

Gen

$$\frac{\Gamma \vdash t : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash t : \forall \bar{\alpha}.\, \sigma}$$

App-Rho

$$\frac{\Gamma \vdash t_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash t_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash t_1\ (t_2 : \exists \bar{\beta}.\, \sigma_2) : \rho}$$

Fun

$$\frac{\Gamma, z : \sigma_2 \vdash t : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (z)\ t : \sigma_2 \to \sigma_1}$$

Let

$$\frac{\Gamma \vdash t_1 : \forall \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, x : \forall \bar{\alpha}.\, \sigma_1[\bar{\tau}/\bar{\beta}] \vdash t_2 : \sigma_2}{\Gamma \vdash \mathsf{let}\ z = (t_1 : \exists \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ t_2 : \sigma_2}$$

Change σ_2 into ρ_2.

$\mathsf{Let}(D_1, D_2) \leadsto \mathsf{Gen}(\mathsf{Let}(D_1, \mathsf{Inst}(D_2)))$

Typing rules for $F_p^{\Downarrow}$

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\Downarrow} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall \bar{\alpha}. \sigma}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists \bar{\beta}. \sigma_2) : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma_1[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists \bar{\beta}. \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2}$$

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \rightarrow \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\sigma_1[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2}$$

Gen remains (it disappears in ML)

May be used in the premisse of Fun and the left premisse of Let.

Var-Inst

$$\frac{\mathsf{x} : \sigma' \in \Gamma \qquad \sigma' \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_1[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma_1[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2}$$

Focus on Let-Gen.

- Remove Distrib
- Distrib-Right is unsound with side effects.
- Distrib makes $\leq^\eta$ undecidable. *Is $\leq_p^\eta$ decidable?*
- $\leq_p^{\Downarrow}$ is decidable (and efficiently computable).
- Thus $\leq_p^{\Downarrow}$ is a strict subrelation of $\leq_p^\eta$.

  $\mathsf{F}_p^{\Downarrow}$ is a less expressive than F_p^η.
- Equivalent simple set of rules for F_p^η:

Inst-Refl
$$\sigma \leqslant \sigma$$

Inst-Arrow
$$\frac{\sigma_1 \leqslant \sigma_1' \qquad \sigma_2' \leqslant \sigma_2}{\sigma_2 \to \sigma_1 \leqslant \sigma_2' \to \sigma_1'}$$

Inst-Skol
$$\frac{\sigma \leqslant \sigma' \qquad \alpha \notin \mathsf{ftv}(\sigma)}{\sigma \leqslant \forall\,\alpha.\,\sigma'}$$

Inst-Spec
$$\frac{\sigma[\tau/\alpha] \leqslant \sigma'}{\forall\,\alpha.\,\sigma \leqslant \sigma'}$$

Type inference via constraints

$$
\begin{aligned}
[\![\mathsf{x} : \rho]\!] &\longrightarrow \mathsf{x} \preceq \rho \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha]\!] &\longrightarrow \exists\, \beta_1 \beta_2 . ([\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \to \beta_2]\!] \wedge \beta_1 \to \beta_2 \leqslant \alpha) \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ [\![\mathsf{t} : \sigma_1]\!] \\
[\![\mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta .\, \sigma_2) : \rho_1]\!] &\longrightarrow \exists\, \bar\beta . ([\![\mathsf{t}_1 : \sigma_2 \to \rho_1]\!] \wedge [\![\mathsf{t}_2 : \sigma_2]\!]) \\
[\![\mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta .\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \forall \bar\beta [[\![\mathsf{t}_1 : \sigma_1]\!]] . \sigma_1\ \mathsf{in}\ [\![\mathsf{t}_2 : \rho_2]\!] \\
[\![\mathsf{t} : \forall\, \bar\alpha .\, \rho]\!] &\longrightarrow \forall\, \bar\alpha . [\![\mathsf{t}_2 : \rho]\!]
\end{aligned}
$$

Or pick any other straightforward type inference algorithm! ▷

$$
\begin{aligned}
[\![\mathsf{x} : \rho]\!] &\longrightarrow \mathsf{x} \preceq \rho \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha]\!] &\longrightarrow \exists\, \beta_1 \beta_2 . ([\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \to \beta_2]\!] \wedge \beta_1 \to \beta_2 \leqslant \alpha) \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ [\![\mathsf{t} : \sigma_1]\!] \\
[\![\mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta . \sigma_2) : \rho_1]\!] &\longrightarrow \exists\, \bar\beta . ([\![\mathsf{t}_1 : \sigma_2 \to \rho_1]\!] \wedge [\![\mathsf{t}_2 : \sigma_2]\!]) \\
[\![\mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta . \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \forall \bar\beta [[\![\mathsf{t}_1 : \sigma_1]\!]] . \sigma_1\ \mathsf{in}\ [\![\mathsf{t}_2 : \rho_2]\!] \\
[\![\mathsf{t} : \forall\, \bar\alpha . \rho]\!] &\longrightarrow \forall\, \bar\alpha . [\![\mathsf{t}_2 : \rho]\!]
\end{aligned}
$$

Logical interpretation of constraints

- Standard interpretation of $\exists$, $\forall$, $\wedge$.
- let constraints can be understood by macro expansion.
- $(\forall \bar\beta [C] . \sigma) \preceq \sigma'$ then means $\exists\, \bar\beta . (C \wedge \sigma \leqslant \sigma')$
- $\leqslant$ constraints are interpreted by $\leq_p^{\shortparallel}$

$$\begin{aligned}
\tau \leqslant \tau' &\longrightarrow \tau = \tau' \\
\sigma_1 \to \sigma_2 \leqslant \sigma_1' \to \sigma_2' &\longrightarrow \sigma_1' \leqslant \sigma_1 \wedge \sigma_2 \to \sigma_2' \\
\forall\,\alpha.\,\sigma \leqslant \rho &\longrightarrow \exists\,\alpha.(\sigma \leqslant \rho) \\
\sigma \leqslant \forall\,\alpha.\,\sigma' &\longrightarrow \forall\,\alpha.(\sigma \leqslant \sigma')
\end{aligned}$$

Follows syntax-directed rules for $\leq_p^{\shortparallel}$

Logical interpretation of constraints

- Standard interpretation of $\exists$, $\forall$, $\wedge$.
- let constraints can be understood by macro expansion.
- $(\forall\bar\beta[C].\sigma) \preceq \sigma'$ then means $\exists\,\bar\beta.(C \wedge \sigma \leqslant \sigma')$
- $\leqslant$ constraints are interpreted by $\leq_p^{\shortparallel}$

$$\llbracket \mathsf{x} : \rho \rrbracket \;\longrightarrow\; \mathsf{x} \preceq \rho$$

$$\llbracket \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha \rrbracket \;\longrightarrow\; \exists\, \beta_1 \beta_2 . (\llbracket \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \to \beta_2 \rrbracket \wedge \beta_1 \to \beta_2 \leqslant \alpha)$$

$$\llbracket \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1 \rrbracket \;\longrightarrow\; \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ \llbracket \mathsf{t} : \sigma_1 \rrbracket$$

$$\llbracket \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta . \sigma_2) : \rho_1 \rrbracket \;\longrightarrow\; \exists\, \bar\beta . (\llbracket \mathsf{t}_1 : \sigma_2 \to \rho_1 \rrbracket \wedge \llbracket \mathsf{t}_2 : \sigma_2 \rrbracket)$$

$$\llbracket \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta . \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2 \rrbracket \;\longrightarrow\; \mathsf{let}\ \mathsf{z} : \forall \bar\beta [\llbracket \mathsf{t}_1 : \sigma_1 \rrbracket] . \sigma_1\ \mathsf{in}\ \llbracket \mathsf{t}_2 : \rho_2 \rrbracket$$

$$\llbracket \mathsf{t} : \forall\, \bar\alpha . \rho \rrbracket \;\longrightarrow\; \forall\, \bar\alpha . \llbracket \mathsf{t}_2 : \rho \rrbracket$$

Logical interpretation of constraints

- Standard interpretation of $\exists$, $\forall$, $\wedge$.
- let constraints can be understood by macro expansion.
- $(\forall \bar\beta [C] . \sigma) \preceq \sigma'$ then means $\exists\, \bar\beta . (C \wedge \sigma \leqslant \sigma')$
- $\leqslant$ constraints are interpreted by $\leq_p^{\shortparallel}$

Type inference via constraints

$$
\begin{aligned}
[\![\mathsf{x} : \rho]\!] &\longrightarrow \mathsf{x} \preceq \rho \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha]\!] &\longrightarrow \exists\, \beta_1\beta_2.([\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \to \beta_2]\!] \wedge \beta_1 \to \beta_2 \leqslant \alpha) \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ [\![\mathsf{t} : \sigma_1]\!] \\
[\![\mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta.\, \sigma_2) : \rho_1]\!] &\longrightarrow \exists\, \bar\beta.([\![\mathsf{t}_1 : \sigma_2 \to \rho_1]\!] \wedge [\![\mathsf{t}_2 : \sigma_2]\!]) \\
[\![\mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \forall \bar\beta[[\![\mathsf{t}_1 : \sigma_1]\!]].\sigma_1\ \mathsf{in}\ [\![\mathsf{t}_2 : \rho_2]\!] \\
[\![\mathsf{t} : \forall\, \bar\alpha.\, \rho]\!] &\longrightarrow \forall\, \bar\alpha.[\![\mathsf{t}_2 : \rho]\!]
\end{aligned}
$$

Inference is first order

- No meta variables for σ or ρ, only for τ.
- Polymorphic shapes are only checked.

$$
\begin{aligned}
[\![x : \rho]\!] &\longrightarrow x \preceq \rho \\
[\![\mathsf{fun}\ (z)\ t : \alpha]\!] &\longrightarrow \exists\, \beta_1\beta_2.([\![\mathsf{fun}\ (z)\ t : \beta_1 \to \beta_2]\!] \wedge \beta_1 \to \beta_2 \leqslant \alpha) \\
[\![\mathsf{fun}\ (z)\ t : \sigma_2 \to \sigma_1]\!] &\longrightarrow \mathsf{let}\ z : \sigma_2\ \mathsf{in}\ [\![t : \sigma_1]\!] \\
[\![t_1\ (t_2 : \exists\, \bar\beta.\, \sigma_2) : \rho_1]\!] &\longrightarrow \exists\, \bar\beta.([\![t_1 : \sigma_2 \to \rho_1]\!] \wedge [\![t_2 : \sigma_2]\!]) \\
[\![\mathsf{let}\ z = (t_1 : \exists\, \bar\beta.\, \sigma_1)\ \mathsf{in}\ t_2 : \rho_2]\!] &\longrightarrow \mathsf{let}\ z : \forall \bar\beta[[\![t_1 : \sigma_1]\!]].\sigma_1\ \mathsf{in}\ [\![t_2 : \rho_2]\!] \\
[\![t : \forall\, \bar\alpha.\, \rho]\!] &\longrightarrow \forall\, \bar\alpha.[\![t_2 : \rho]\!]
\end{aligned}
$$

Type inference problem

$\Gamma \ \vdash t : \quad \sigma$ is solved as

$\mathsf{let}\ \Gamma\ \mathsf{in}\ [\![t : \sigma]\!]$

$$
\begin{aligned}
[\![\mathsf{x} : \rho]\!] &\longrightarrow \mathsf{x} \preceq \rho \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha]\!] &\longrightarrow \exists\, \beta_1 \beta_2 . ([\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \to \beta_2]\!] \wedge \beta_1 \to \beta_2 \leqslant \alpha) \\
[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ [\![\mathsf{t} : \sigma_1]\!] \\
[\![\mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar\beta . \sigma_2) : \rho_1]\!] &\longrightarrow \exists\, \bar\beta . ([\![\mathsf{t}_1 : \sigma_2 \to \rho_1]\!] \wedge [\![\mathsf{t}_2 : \sigma_2]\!]) \\
[\![\mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar\beta . \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2]\!] &\longrightarrow \mathsf{let}\ \mathsf{z} : \forall \bar\beta [[\![\mathsf{t}_1 : \sigma_1]\!]] . \sigma_1\ \mathsf{in}\ [\![\mathsf{t}_2 : \rho_2]\!] \\
[\![\mathsf{t} : \forall\, \bar\alpha . \rho]\!] &\longrightarrow \forall\, \bar\alpha . [\![\mathsf{t}_2 : \rho]\!]
\end{aligned}
$$

Type inference problem

find φ such that $\varphi(\Gamma) \vdash \mathsf{t} : \varphi(\sigma)$ is solved as

find φ such that $\varphi \Vdash \mathsf{let}\ \Gamma\ \mathsf{in}\ [\![\mathsf{t} : \sigma]\!]$

This algorithm is sound and complete (proof to be done).

Type inference via constraints

$$[\![\mathsf{x} : \rho]\!] \longrightarrow \mathsf{x} \preceq \rho$$

$$[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \alpha]\!] \longrightarrow \exists\, \beta_1 \beta_2 . ([\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \beta_1 \rightarrow \beta_2]\!] \wedge \beta_1 \rightarrow \beta_2 \leqslant \alpha)$$

$$[\![\mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1]\!] \longrightarrow \mathsf{let}\ \mathsf{z} : \sigma_2\ \mathsf{in}\ [\![\mathsf{t} : \sigma_1]\!]$$

$$[\![\mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \rho_1]\!] \longrightarrow \exists\, \bar{\beta}.([\![\mathsf{t}_1 : \sigma_2 \rightarrow \rho_1]\!] \wedge [\![\mathsf{t}_2 : \sigma_2]\!])$$

$$[\![\mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_2]\!] \longrightarrow \mathsf{let}\ \mathsf{z} : \forall \bar{\beta}[[\![\mathsf{t}_1 : \sigma_1]\!]].\sigma_1\ \mathsf{in}\ [\![\mathsf{t}_2 : \rho_2]\!]$$

$$[\![\mathsf{t} : \forall\, \bar{\alpha}.\, \rho]\!] \longrightarrow \forall\, \bar{\alpha}.[\![\mathsf{t}_2 : \rho]\!]$$

In the presence of side effects?

Value restriction

Restrict generalization to values or non expansive expressions?

Is it sound?

- Recall Distrib is not sound with side effects.
  (would allow: $\forall\,\alpha.\,\mathsf{unit} \to \mathsf{ref}\,\alpha \le \mathsf{unit} \to \forall\,\alpha.\,\mathsf{ref}\,\alpha$)
- $(\le_p^{\shortparallel})$ does not include Distrib (it is incomplete).
- It is sound because $(\le_p^{\shortparallel}) \subseteq (\le_v^{\eta})$.

Is type inference complete?

No, it must be modified! Applying Gen only at the end is not sufficient. Change rules App and Let:

App$_v$

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2[\bar\tau/\bar\beta] \to \sigma_1 \qquad \Gamma \vdash \mathsf{t}_2 : \sigma_2[\bar\tau/\bar\beta] \qquad \mathsf{t}_1\;\mathsf{t}_2 \notin \mathcal{U}}{\Gamma \vdash \mathsf{t}_1\;(\mathsf{t}_2 : \exists\,\bar\beta.\,\sigma_2) : \sigma_1}$$

Our slogan, once again

Type inference infers monotypes but only checks polytypes!

Can we make this formal? and exploit it better?

Shapes $\mathcal{S}$

- We extend polytypes with a constant $\sharp$ to represent monotypes.
- Shapes are closed polytypes with $\sharp$
  (free variables are monotypes represented by $\sharp$)
- Shapes are taken modulo $\sharp \to \sharp = \sharp$
  (we ignore the structure of monotypes)

Shape checking

Our slogan, once again

Type inference infers monotypes but only checks polytypes!

Can we make this formal? and exploit it better?

Shapes $\mathcal{S}$: closed polytypes with a constant $\sharp$ and $\sharp = \sharp \rightarrow \sharp$

Operations on shapes

$\lceil \sigma \rceil$ returns the shape of σ, i.e. $\sigma[\sharp/\mathsf{ftv}(\sigma)]$

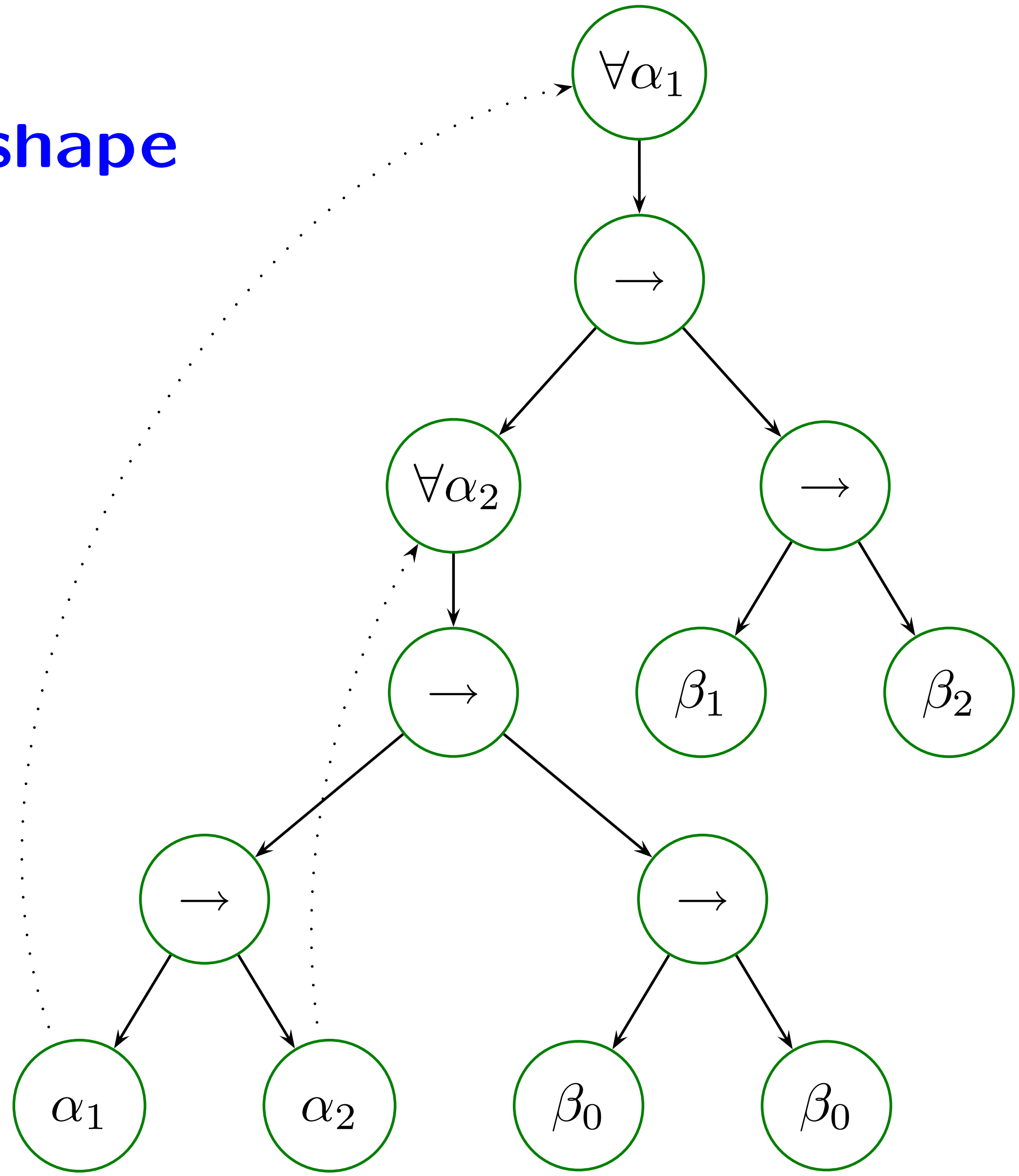
$\mathcal{S}^\flat$ strips $\mathcal{S}$ off its toplevel quantifiers and reshape (replace free variables by $\sharp$).

We write $\mathcal{R}$ for stripped shapes.

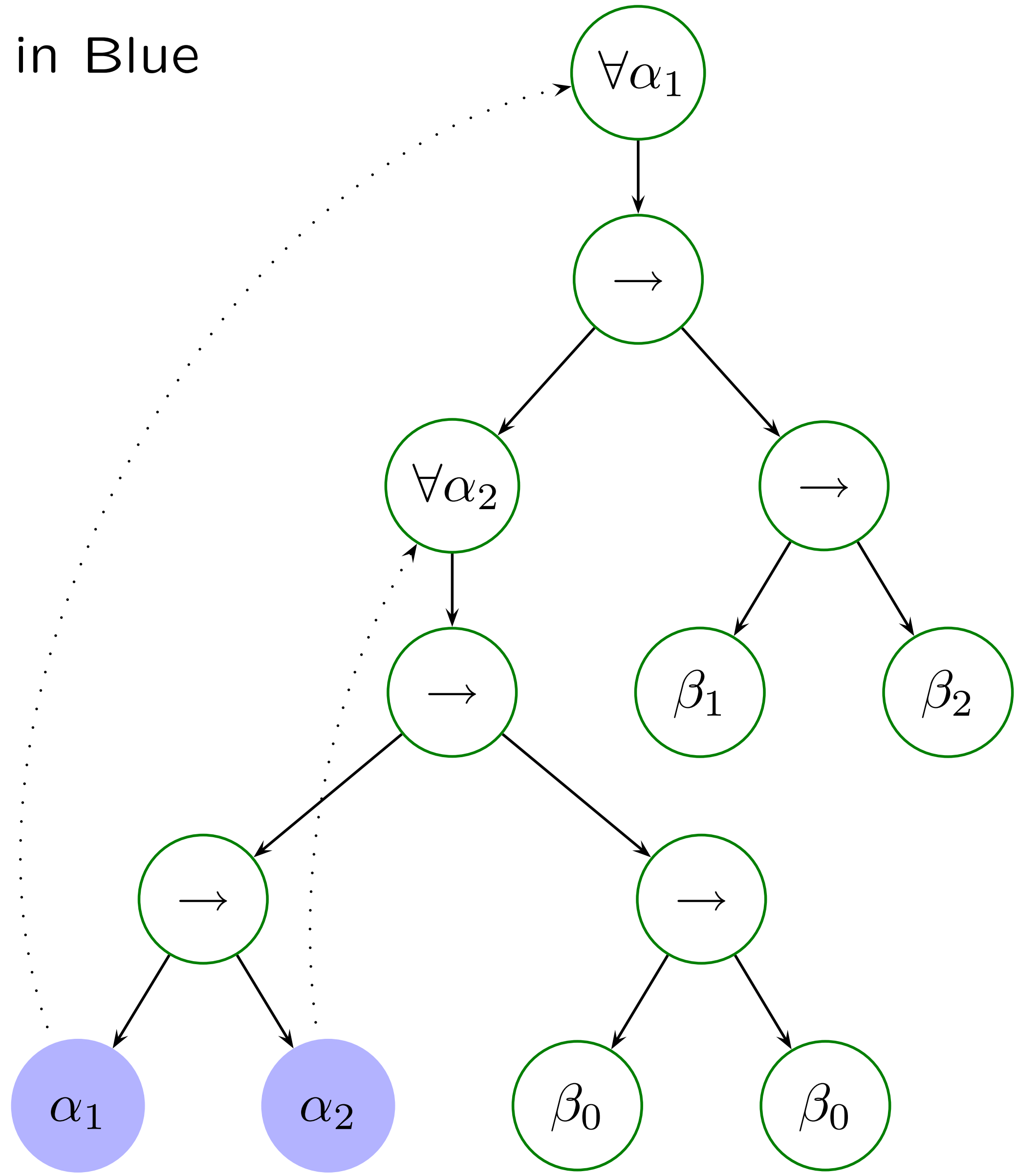
$\lfloor \mathcal{S} \rfloor$ returns the annotation $\exists\, \bar{\beta}.\, \mathcal{S}[\beta_i/\sharp_i]$.

▷

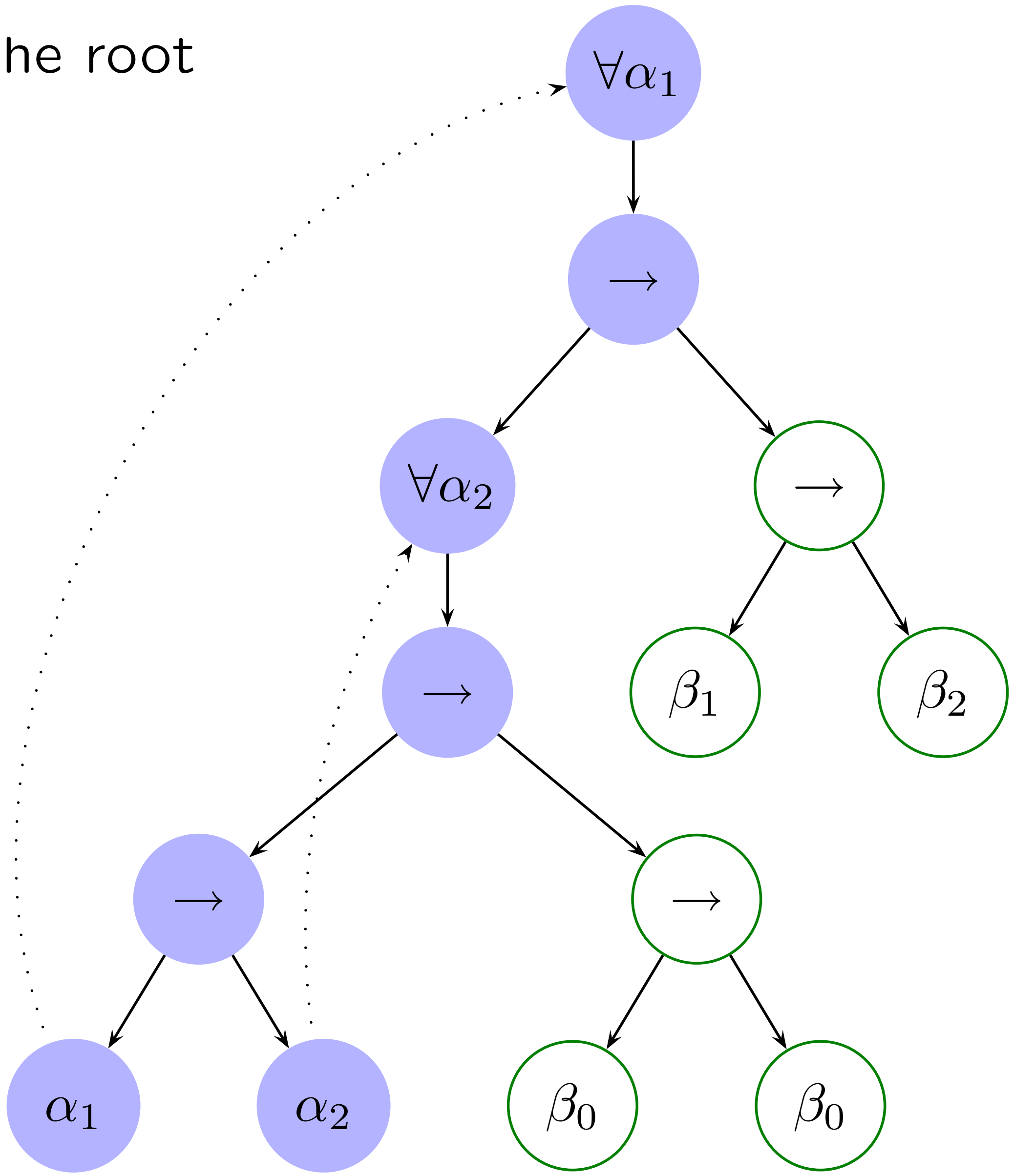
Computing the shape



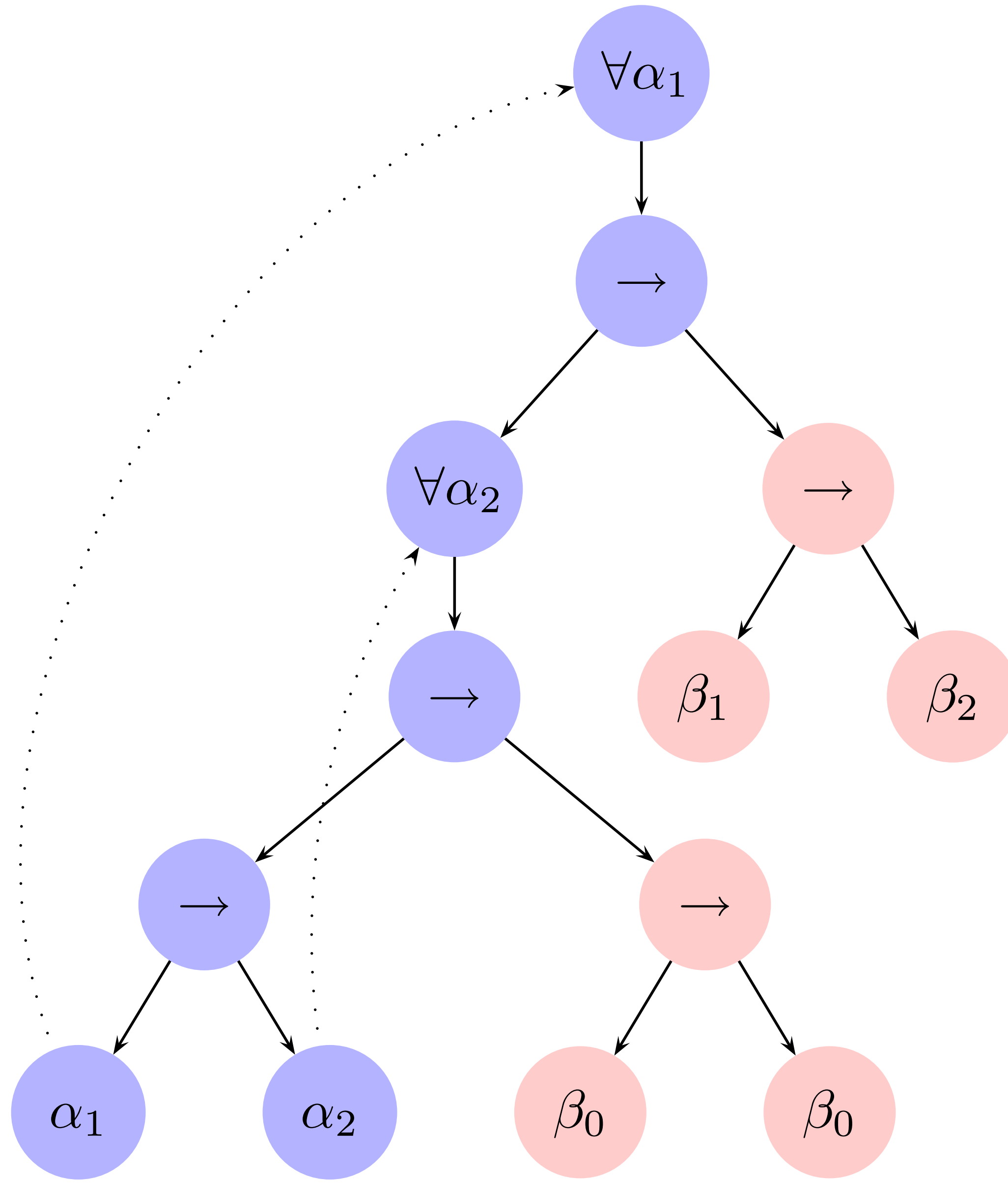
▷ Mark bound variables in Blue



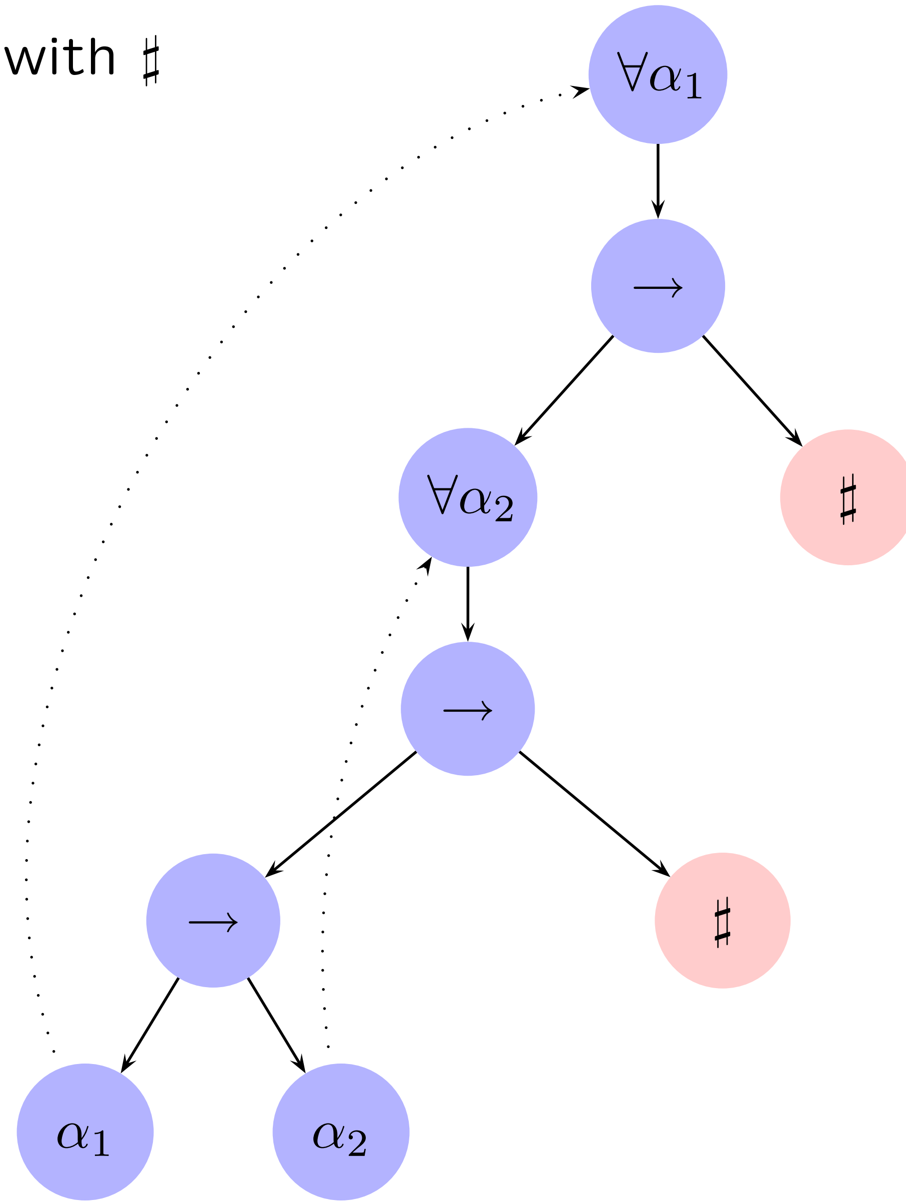
▷ Spread blue towards the root



▷ Mark the rest in red



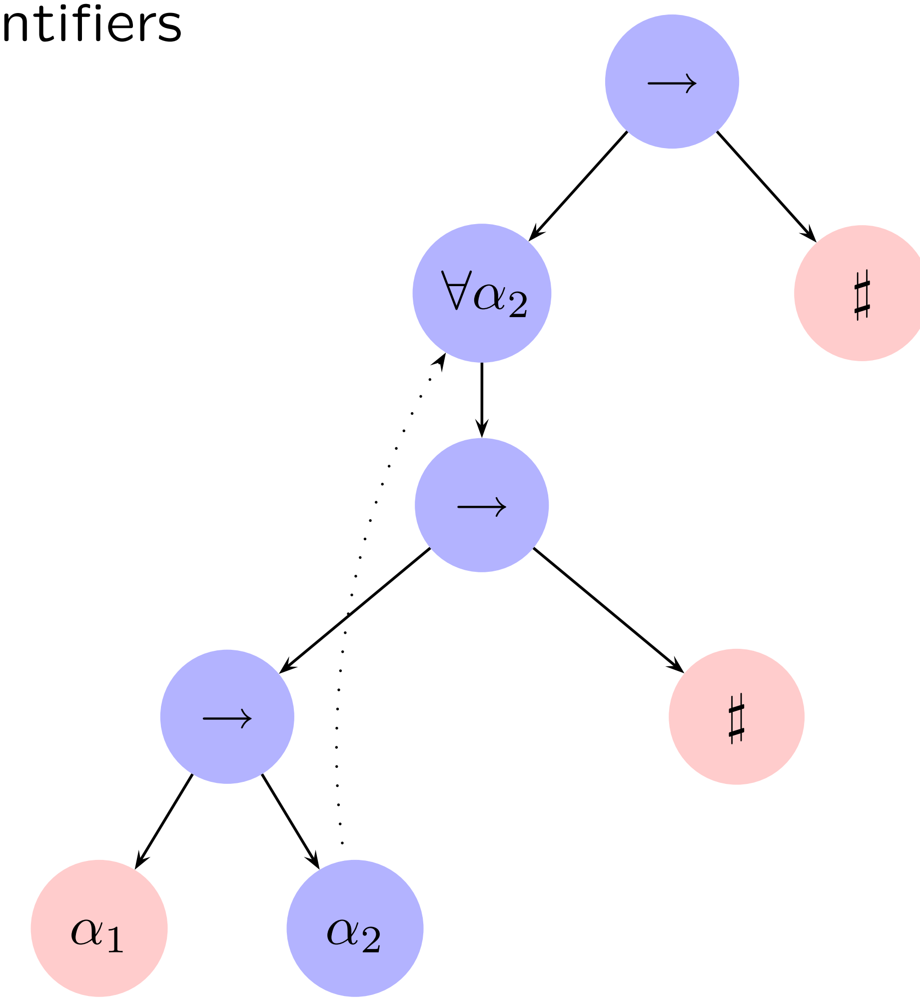
▷ Replace red subtrees with $\sharp$



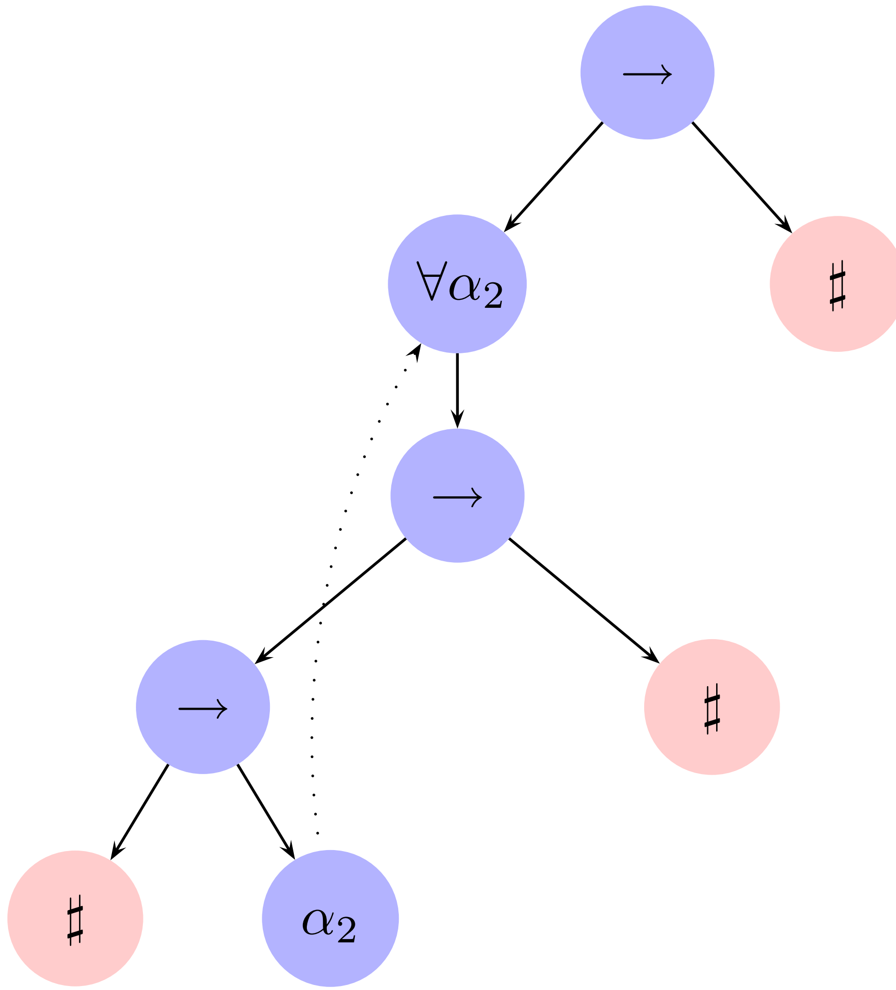
▷

Stripping a shape

Remove toplevel quantifiers



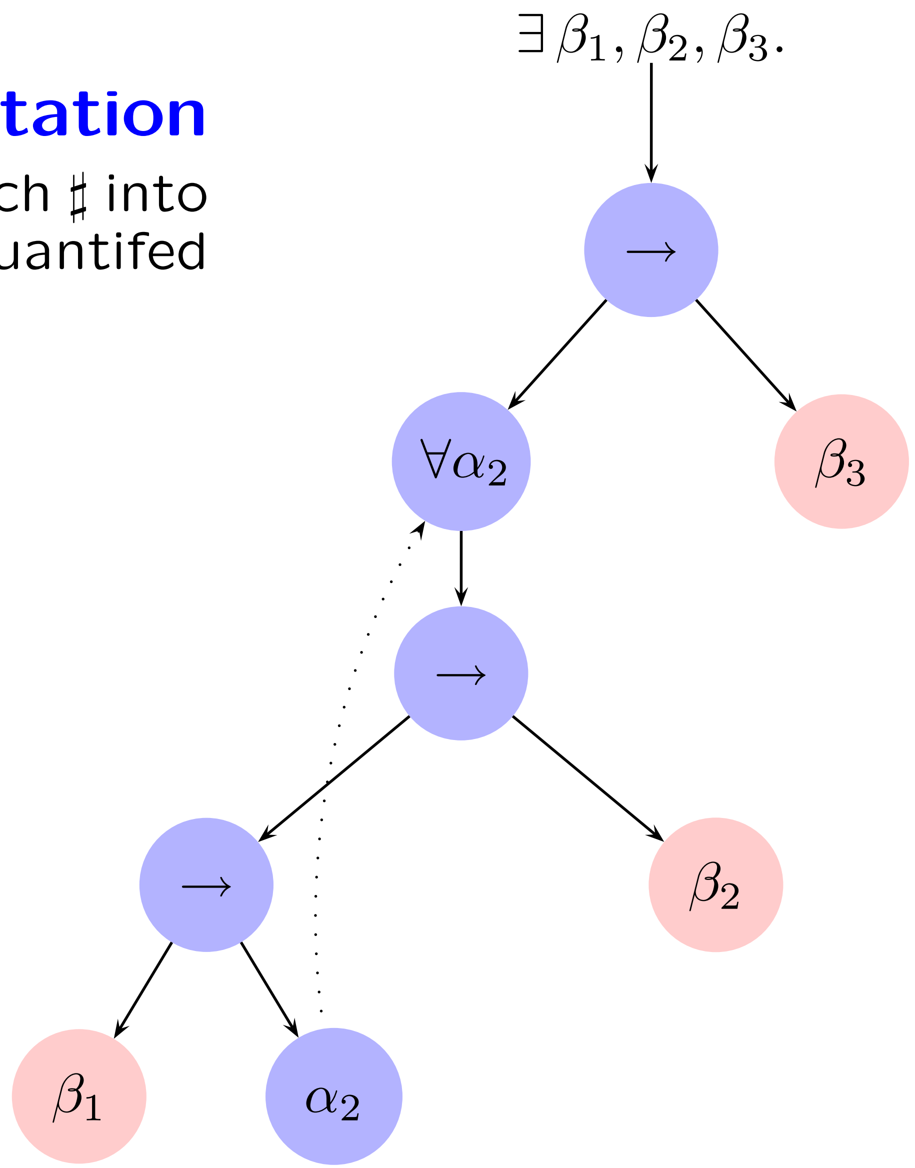
▷ Reshape



▷

Building an annotation

from a shape: turn each $\sharp$ into a fresh existentially quantifed variable.



$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

Var-Rho

$$\frac{\mathsf{x} : \sigma \in \Gamma \qquad \sigma \leq_{p}^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Gen

$$\frac{\Gamma \vdash \mathsf{t} : \sigma \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma}$$

Var-Rho

$$\frac{\mathsf{x} : \sigma \in \Gamma \qquad \sigma \leq^{\shortparallel}_{p} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \rightarrow \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \sigma \in \Gamma \qquad \sigma \leq^{\shortparallel}_{p} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \rightarrow \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \sigma \in \Gamma \qquad \sigma \leq_p^{\shortparallel} \rho}{\Gamma \vdash \mathsf{x} : \rho}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \rightarrow \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{!!} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\,\bar{\beta}.\,\sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\,\bar{\beta}.\,\sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash \mathsf{t} : \sigma_1}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \to \sigma_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^{\flat} \vdash \mathsf{t} : \mathcal{S}_1^{\flat}}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^{\flat} \vdash \mathsf{t} : \mathcal{S}_1^{\flat}}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \to \rho \qquad \Gamma \vdash \mathsf{t}_2 : \sigma[\bar{\tau}/\bar{\beta}]}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \rho}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^{\flat} \vdash \mathsf{t} : \mathcal{S}_1^{\flat}}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \rightarrow \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma[\bar{\tau}/\bar{\beta}] \rceil \rightarrow \mathcal{R} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma[\bar{\tau}/\bar{\beta}] \rceil^{\flat}}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists \bar{\beta}. \sigma) : \mathcal{R}}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists \bar{\beta}. \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^\flat \vdash \mathsf{t} : \mathcal{S}_1^\flat}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil \to \mathcal{S} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma \rceil^\flat}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \mathcal{S}^\flat}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

Shape checking rules

$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{!!} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^\flat \vdash \mathsf{t} : \mathcal{S}_1^\flat}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil \to \mathcal{S} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma \rceil^\flat}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \mathcal{S}^\flat}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma[\bar{\tau}/\bar{\beta}] \qquad \Gamma, \mathsf{z} : \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \vdash \mathsf{t}_2 : \rho}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \rho}$$

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^\flat \vdash \mathsf{t} : \mathcal{S}_1^\flat}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil \to \mathcal{S} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma \rceil^\flat}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \mathcal{S}^\flat}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma[\bar{\tau}/\bar{\beta}] \rceil^\flat \qquad \Gamma, \mathsf{z} : \lceil \langle \Gamma \rangle (\sigma[\bar{\tau}/\bar{\beta}]) \rceil^\flat \vdash \mathsf{t}_2 : \mathcal{R}}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^\flat \vdash \mathsf{t} : \mathcal{S}_1^\flat}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \rightarrow \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil \rightarrow \mathcal{S} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma \rceil^\flat}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \mathcal{S}^\flat}$$

Let-Gen-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil^\flat \qquad \Gamma, \mathsf{z} : \lceil \sigma \rceil^\flat \vdash \mathsf{t}_2 : \mathcal{R}}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}}$$

Shape checking rules

$$\Gamma \vdash_{\Downarrow} \mathsf{t} : \mathcal{R}$$

Var-Rho

$$\frac{\mathsf{x} : \mathcal{R}' \in \Gamma \qquad \mathcal{R}' \leq_p^{\shortparallel} \mathcal{R}}{\Gamma \vdash \mathsf{x} : \mathcal{R}}$$

Fun

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^\flat \vdash \mathsf{t} : \mathcal{S}_1^\flat}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

App-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil \to \mathcal{S} \qquad \Gamma \vdash \mathsf{t}_2 : \lceil \sigma \rceil^\flat}{\Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma) : \mathcal{S}^\flat}$$

Let-Gen-Rho

$$\frac{\Gamma \vdash \mathsf{t}_1 : \lceil \sigma \rceil^\flat \qquad \Gamma, \mathsf{z} : \lceil \sigma \rceil^\flat \vdash \mathsf{t}_2 : \mathcal{R}}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}}$$

Properties of shape checking

Well-typed programs are well-shaped

If $\Gamma \vdash \mathsf{t} : \sigma$ then $\lceil\Gamma\rceil \vdash_{\Downarrow} \mathsf{t} : \lceil\sigma\rceil$.

Only shapes of annotations matters

If $\Gamma \vdash \mathsf{t} : \sigma$ then $\Gamma \vdash \lfloor\lceil\mathsf{t}\rceil\rfloor : \sigma$.

Shape checking

Shape information flows downward (from root towards leaves)

Shape inferrence

Let shape information flow upward (from leaves to the root)

We need annotations at different places...

$$\mathsf{t} ::= \mathsf{x} \;\mid\; \mathsf{fun}\,(\mathsf{z} : \theta)\;\mathsf{t} \;\mid\; \mathsf{t}_1\;\mathsf{t}_2 \;\mid\; \mathsf{let}\;\mathsf{z} = \mathsf{t}_1\;\mathsf{in}\;\mathsf{t}_2$$

Shape inference

$$\Gamma \vdash_{\Uparrow} \mathsf{t} : \mathcal{R}$$

Var-I

$$\frac{\mathsf{x} : \mathcal{R} \in \Gamma}{\Gamma \vdash_{\Uparrow} \mathsf{x} : \mathcal{R}}$$

App-I

$$\frac{\Gamma \vdash_{\Uparrow} \mathsf{t}_1 : \mathcal{S}_2 \rightarrow \mathcal{S}_1 \qquad \Gamma \vdash_{\Uparrow} \mathsf{t}_2 : \mathcal{R}_2 \qquad \mathcal{R}_2 \leq_p^{\shortparallel} \mathcal{S}_2^{\flat}}{\Gamma \vdash_{\Uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{S}_1^{\flat}}$$

Let-I

$$\frac{\Gamma \vdash_{\Uparrow} \mathsf{t}_1 : \mathcal{R}_1 \qquad \Gamma, \mathsf{z} : \mathcal{R}_1 \vdash_{\Uparrow} \mathsf{t}_2 : \mathcal{R}_2}{\Gamma \vdash_{\Uparrow} \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}_2}$$

Fun-I

$$\frac{\Gamma, \mathsf{z} : \lceil \sigma \rceil^{\flat} \vdash_{\Uparrow} \mathsf{t} : \mathcal{R}}{\Gamma \vdash_{\Uparrow} \mathsf{fun}\ (\mathsf{z} : \exists \bar{\beta}. \sigma)\ \mathsf{t} : \lceil \sigma \rceil \rightarrow \mathcal{R}}$$

Shape inference

$$\Gamma \vdash_{\Uparrow} \mathsf{t} : \mathcal{R} \Rightarrow \mathsf{t}'$$

Var-I

$$\frac{\mathsf{x} : \mathcal{R} \in \Gamma}{\Gamma \vdash_{\Uparrow} \mathsf{x} : \mathcal{R} \Rightarrow \mathsf{x}}$$

App-I

$$\frac{\Gamma \vdash_{\Uparrow} \mathsf{t}_1 : \mathcal{S}_2 \to \mathcal{S}_1 \Rightarrow \mathsf{t}'_1 \qquad \Gamma \vdash_{\Uparrow} \mathsf{t}_2 : \mathcal{R}_2 \Rightarrow \mathsf{t}'_2 \qquad \mathcal{R}_2 \leq^{\shortparallel}_{p} \mathcal{S}^{\flat}_2}{\Gamma \vdash_{\Uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{S}^{\flat}_1 \Rightarrow \mathsf{t}'_1\ ((\mathsf{t}'_2 : \lfloor \mathcal{R}_2 \rfloor) : \lfloor \mathcal{S}_2 \rfloor)}$$

Let-I

$$\frac{\Gamma \vdash_{\Uparrow} \mathsf{t}_1 : \mathcal{R}_1 \Rightarrow \mathsf{t}'_1 \qquad \Gamma, \mathsf{z} : \mathcal{R}_1 \vdash_{\Uparrow} \mathsf{t}_2 : \mathcal{R}_2 \Rightarrow \mathsf{t}'_2}{\Gamma \vdash_{\Uparrow} \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}_2 \Rightarrow \mathsf{let}\ \mathsf{z} = (\mathsf{t}'_1 : \lfloor \mathcal{R}_1 \rfloor)\ \mathsf{in}\ \mathsf{t}'_2}$$

Fun-I

$$\frac{\Gamma, \mathsf{z} : \lceil \sigma \rceil^{\flat} \vdash_{\Uparrow} \mathsf{t} : \mathcal{R} \Rightarrow \mathsf{t}'}{\Gamma \vdash_{\Uparrow} \mathsf{fun}\ (\mathsf{z} : \exists \bar{\beta}. \sigma)\ \mathsf{t} : \lceil \sigma \rceil \to \mathcal{R} \Rightarrow \mathsf{fun}\ (\mathsf{z})\ \mathsf{let}\ \mathsf{z} = (\mathsf{z} : \exists \bar{\beta}. \sigma)\ \mathsf{in}\ \mathsf{t}'}$$

Typing $\mathsf{F}_p^{\Uparrow}$ by elaboration into $\mathsf{F}_p^{\Downarrow}$

Define

$$\Gamma \vdash_{\Uparrow} \mathsf{t} : \sigma \iff \exists \mathcal{R},\ (\lceil \Gamma \rceil^{\flat} \vdash_{\Uparrow} \mathsf{t} : \mathcal{R} \Rightarrow \mathsf{t}' \ \wedge\ \mathcal{R} = \sigma^{\flat} \ \wedge\ \Gamma \vdash_{\Downarrow} \mathsf{t} : \sigma \Rightarrow \mathsf{t}')$$

Direct specification of $\mathsf{F}_p^{\Uparrow}$

Var-Inst

$$\frac{\mathsf{x} : \forall\, \bar\alpha.\, \rho \in \Gamma}{\Gamma \vdash \mathsf{x} : \rho[\bar\tau/\bar\alpha]}$$

Fun-Gen

$$\frac{\Gamma, \mathsf{z} : \sigma[\bar\tau/\bar\beta] \vdash \mathsf{t} : \rho}{\Gamma \vdash \mathsf{fun}\ (\mathsf{z} : \exists\, \bar\beta.\, \sigma)\ \mathsf{t} : \sigma[\bar\tau/\bar\beta] \rightarrow \rho}$$

App

$$\frac{\Gamma \vdash \mathsf{t}_1 : \sigma_2 \rightarrow \forall\, \bar\alpha.\, \rho_1 \qquad \Gamma \vdash \mathsf{t}_2 : \rho_2 \qquad \langle\Gamma\rangle(\rho_2) \leq_p^{\shortparallel} \sigma_2}{\Gamma \vdash \mathsf{t}_1\ \mathsf{t}_2 : \rho_1[\bar\tau/\bar\alpha]}$$

Let-Gen

$$\frac{\Gamma \vdash \mathsf{t}_1 : \rho_1 \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\rho_1) \vdash \mathsf{t}_2 : \rho_2}{\Gamma \vdash \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \rho_2}$$

Canonical types

No quantifier immediately on the right of an arrow

$$\rho ::= \tau \mid \sigma \to \rho \qquad \text{rho type}$$

$$\sigma ::= \rho \mid \forall\, \alpha.\, \sigma \qquad \text{type scheme}$$

The system $\mathsf{F}_p^{\Uparrow}$ is then equivalent to OL's system.

Allow non canonical types?

They are useless in F^{η}, since

$$\forall\,\bar\alpha.\,\sigma \to \rho \equiv^{\eta} \sigma \to \forall\,\bar\alpha.\,\rho \qquad \bar\alpha \notin \mathsf{ftv}(\sigma)$$

However, this is no more true in F^{η}_{v} since

$$\forall\,\bar\alpha.\,\sigma \to \rho \not\leq^{\eta}_{v} \sigma \to \forall\,\bar\alpha.\,\rho \qquad \bar\alpha \notin \mathsf{ftv}(\sigma)$$

is unsound.

Then, non-canonical types make a difference
(they allow more unanotated programs to be annotated so that they are typable)

Three views in $\mathsf{F}_p^{\Downarrow}/\mathsf{F}_p^{\Uparrow}$

Note that even without side effect

$$\forall\,\bar\alpha.\,\sigma \to \rho \not\preceq_p^{\shortparallel} \sigma \to \forall\,\bar\alpha.\,\rho \qquad \bar\alpha \notin \mathsf{ftv}(\sigma)$$

(1) allow non canonical types.

(2) restrict to canonical types.

(3) restrict to canonical types and canonize input types.

Then $(2) \subset (1) \subset (3)$.

Why is $F_p^{\Uparrow}$ weaker than OL with canonical types?

- During shape inference $F_p^{\Uparrow}$ ignores toplevel quantifiers.

  Because shape ignores variables.

  So in rule Let-Gen it can only tell the best stripped shape.

- OL mixies shape inference and type inference.

  It thus knows variables as well as the shape, and in Let-Gen it can put in the environment the best non-stripped shape.

- This is a small weakness (price to pay?) for a clear separation of shape inference and monotype inference.

In $F_p^{\Downarrow}$, with shape checking only:

$$(\text{fun } (z)\ (\text{fun } (y)\ y : \sigma_{id} \rightarrow \sigma_{id}) : \alpha \rightarrow \sigma_{id} \rightarrow \sigma_{id})$$

Repeating the toplevel blue annotation is needed but annoying.

In $F_p^{\Uparrow}$, with shape inference only:

$$(\text{fun } (z)\ (\text{fun } (y : \sigma)\ y) : \alpha \rightarrow \sigma_{id} \rightarrow \sigma_{id})$$

Repeating the inner blue annotation is needed but annoying.

Can both be mixed?

Var-C

$$\frac{}{\Gamma \vdash_{\downarrow} \mathsf{x} : \mathcal{R}}$$

Var-I

$$\frac{\mathsf{x} : \mathcal{R} \in \Gamma}{\Gamma \vdash_{\uparrow} \mathsf{x} : \mathcal{R}}$$

App-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \mathcal{S}_2 \to \mathcal{S}_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \mathcal{S}_2^{\flat} \qquad \mathcal{S}_1^{\flat} \leq_p^{\shortparallel} \mathcal{R}_1}{\Gamma \vdash_{\downarrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{R}_1}$$

App-I

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \mathcal{S}_2 \to \mathcal{S}_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \mathcal{S}_2^{\flat}}{\Gamma \vdash_{\uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{S}_1^{\flat}}$$

Let-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \mathcal{R}_1 \qquad \Gamma, \mathsf{z} : \mathcal{R}_1 \vdash_{\epsilon} \mathsf{t}_2 : \mathcal{R}_2}{\Gamma \vdash_{\epsilon} \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \mathcal{R}_2}$$

Fun-Ce

$$\frac{\Gamma, \mathsf{z} : \lceil \sigma \rceil^{\flat} \vdash_{\downarrow} \mathsf{t} : \mathcal{S}_1^{\flat} \qquad \mathcal{S}_2^{\flat} \leq_p^{\shortparallel} \lceil \sigma \rceil^{\flat}}{\Gamma \vdash_{\downarrow} \mathsf{fun}\ (\mathsf{z} : \exists\, \beta.\, \sigma)\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

Fun-Ie

$$\frac{\Gamma, \mathsf{z} : \lceil \sigma \rceil^{\flat} \vdash_{\uparrow} \mathsf{t} : \mathcal{R}}{\Gamma \vdash_{\uparrow} \mathsf{fun}\ (\mathsf{z} : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{t} : \lceil \sigma \rceil \to \mathcal{R}}$$

Fun-Ci

$$\frac{\Gamma, \mathsf{z} : \mathcal{S}_2^{\flat} \vdash_{\downarrow} \mathsf{t} : \mathcal{S}_1^{\flat}}{\Gamma \vdash_{\downarrow} \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

Fun-Ii

$$\frac{\Gamma, \mathsf{z} : \sharp \vdash_{\uparrow} \mathsf{t} : \mathcal{R}}{\Gamma \vdash_{\uparrow} \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sharp \to \mathcal{R}}$$

Var-C

Var-I

App-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \mathcal{S}_2 \to \mathcal{S}_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \mathcal{S}_2^{\flat} \qquad \mathcal{S}_1^{\flat} \leq_p^{\shortparallel} \mathcal{R}_1}{\Gamma \vdash_{\downarrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{R}_1}$$

App-I

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \mathcal{S}_2 \to \mathcal{S}_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \mathcal{S}_2^{\flat}}{\Gamma \vdash_{\uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \mathcal{S}_1^{\flat}}$$

Let-C

Fun-Ce

$$\frac{\Gamma, \mathsf{z} : \lceil \sigma \rceil^{\flat} \vdash_{\downarrow} \mathsf{t} : \mathcal{S}_1^{\flat} \qquad \mathcal{S}_2^{\flat} \leq_p^{\shortparallel} \lceil \sigma \rceil^{\flat}}{\Gamma \vdash_{\downarrow} \mathsf{fun}\ (\mathsf{z} : \exists \beta.\, \sigma)\ \mathsf{t} : \mathcal{S}_2 \to \mathcal{S}_1}$$

Fun-Ie

Fun-Ci

Fun-Ii

$$\frac{\Gamma, \mathsf{z} : \sharp \vdash_{\uparrow} \mathsf{t} : \mathcal{R}}{\Gamma \vdash_{\uparrow} \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sharp \to \mathcal{R}}$$

Var-C Var-I

App-C

$$\frac{\Gamma \vdash_{\uparrow} t_1 : \mathcal{S}_2 \to \mathcal{S}_1 \Rightarrow t_1' \qquad \Gamma \vdash_{\downarrow} t_2 : \mathcal{S}_2^{\flat} \Rightarrow t_2' \qquad \mathcal{S}_1^{\flat} \leq_p^{\shortparallel} \mathcal{R}_1}{\Gamma \vdash_{\downarrow} t_1\ t_2 : \mathcal{R}_1 \Rightarrow t_1\ (t_2 : \lfloor \mathcal{S}_2 \rfloor)}$$

App-I

$$\frac{\Gamma \vdash_{\uparrow} t_1 : \mathcal{S}_2 \to \mathcal{S}_1 \Rightarrow t_1' \qquad \Gamma \vdash_{\downarrow} t_2 : \mathcal{S}_2^{\flat} \Rightarrow t_2'}{\Gamma \vdash_{\uparrow} t_1\ t_2 : \mathcal{S}_1^{\flat} \Rightarrow t_1\ (t_2 : \lfloor \mathcal{S}_2 \rfloor)}$$

Let-C

Fun-Ce

$$\frac{\Gamma, z : \lceil \sigma \rceil^{\flat} \vdash_{\downarrow} t : \mathcal{S}_1^{\flat} \Rightarrow t' \qquad \mathcal{S}_2^{\flat} \leq_p^{\shortparallel} \lceil \sigma \rceil^{\flat}}{\Gamma \vdash_{\downarrow} \text{fun}\ (z : \exists\, \beta.\, \sigma)\ t : \mathcal{S}_2 \to \mathcal{S}_1 \Rightarrow \text{fun}\ (z)\ \text{let}\ z = (z : \exists\, \beta.\, \sigma)\ \text{in}\ t'}$$

Fun-Ie Fun-Ci

Fun-Ii

$$\frac{\Gamma, z : \sharp \vdash_{\uparrow} t : \mathcal{R} \Rightarrow t'}{\Gamma \vdash_{\uparrow} \text{fun}\ (z)\ t : \sharp \to \mathcal{R} \Rightarrow \text{fun}\ (z)\ t'}$$

Var-C

$$\frac{\sigma' \in \Gamma \qquad \sigma' \leq_p^{''} \rho}{\Gamma \vdash_\downarrow \mathsf{z} : \rho}$$

Var-I

$$\frac{\mathsf{z} : \sigma \in \Gamma \qquad \sigma \leq_p \rho}{\Gamma \vdash_\uparrow \mathsf{z} : \rho}$$

App-C

$$\frac{\Gamma \vdash_\uparrow \mathsf{t}_1 : \sigma_2 \rightarrow \sigma_1 \quad \Gamma \vdash_\downarrow \mathsf{t}_2 : \sigma_2 \qquad \langle\Gamma\rangle(\sigma_1) \leq_p^{''} \rho_1}{\Gamma \vdash_\downarrow \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

App-I

$$\frac{\Gamma \vdash_\uparrow \mathsf{t}_1 : \sigma_2 \rightarrow \rho_1 \quad \Gamma \vdash_\downarrow \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_\uparrow \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

Let-C

$$\frac{\Gamma \vdash_\uparrow \mathsf{t}_1 : \rho_1 \quad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\rho_1) \vdash_\downarrow \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_\downarrow \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Let-I

$$\frac{\Gamma \vdash_\uparrow \mathsf{t}_1 : \rho_1 \quad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\rho_1) \vdash_\uparrow \mathsf{t}_2 : \rho_2}{\Gamma \vdash_\uparrow \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \rho_2}$$

Fun-Ci

$$\frac{\Gamma, \mathsf{z} : \sigma_2 \vdash_\downarrow \mathsf{t} : \sigma_1}{\Gamma \vdash_\downarrow \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma_2 \rightarrow \sigma_1}$$

Fun-Ie

$$\frac{\Gamma, \mathsf{z} : \sigma[\tau/\bar{\beta}] \vdash_\uparrow \mathsf{t} : \rho}{\Gamma \vdash_\uparrow \mathsf{fun}\ (\mathsf{z} : \exists\, \bar{\beta}.\, \sigma)\ \mathsf{t} : \sigma[\tau/\bar{\beta}] \rightarrow \rho}$$

Fun-Ce

$$\frac{\Gamma, \mathsf{z} : \sigma[\bar{\tau}/\bar{\beta}] \vdash_\downarrow \mathsf{t} : \sigma_1 \qquad \sigma_2 \leq_p^{''} \sigma[\bar{\tau}/\bar{\beta}] \qquad \bar{\alpha} \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash_\downarrow \mathsf{fun}\ (\mathsf{z} : \exists\, \beta.\, \sigma)\ \mathsf{t} : \forall\, \bar{\alpha}.\, \sigma_2 \rightarrow \sigma_1}$$

Fun-Ii

$$\frac{\Gamma, \mathsf{z} : \tau \vdash_\uparrow \mathsf{t} : \rho}{\Gamma \vdash_\uparrow \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau \rightarrow \rho}$$

---

[a] Quite simple!Proof: (by Simon) Only requires xx new linesto the Haskell typechecker:-)

Bidirectional type inference $F_p^{\Updownarrow}$

App-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \sigma_2 \to \sigma_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2 \qquad \langle\Gamma\rangle(\sigma_1) \leq_p^{\shortparallel} \rho_1}{\Gamma \vdash_{\downarrow} \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

App-I

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \sigma_2 \to \rho_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_{\uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

Var-C

Var-I

Let-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \rho_1 \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\rho_1) \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_{\downarrow} \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Let-I

Fun-Ci

Fun-Ie

Fun-Ce

$$\frac{\Gamma, \mathsf{z} : \sigma[\bar\tau/\bar\beta] \vdash_{\downarrow} \mathsf{t} : \sigma_1 \qquad \sigma_2 \leq_p^{\shortparallel} \sigma[\bar\tau/\bar\beta] \qquad \bar\alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash_{\downarrow} \mathsf{fun}\ (\mathsf{z} : \exists\,\beta.\,\sigma)\ \mathsf{t} : \forall\,\bar\alpha.\,\sigma_2 \to \sigma_1}$$

Fun-Ii

$$\frac{\Gamma, \mathsf{z} : \tau \vdash_{\uparrow} \mathsf{t} : \rho}{\Gamma \vdash_{\uparrow} \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau \to \rho}$$

Bidirectional type inference $F_p^{\Updownarrow}$

Var-C

Var-I

App-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \sigma_2 \to \sigma_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2 \qquad \langle\Gamma\rangle(\sigma_1) \leq_p^{\shortparallel} \rho_1}{\Gamma \vdash_{\downarrow} \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

App-I

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \sigma_2 \to \rho_1 \qquad \Gamma \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_{\uparrow} \mathsf{t}_1\ \mathsf{t}_2 : \rho_1}$$

Let-C

$$\frac{\Gamma \vdash_{\uparrow} \mathsf{t}_1 : \rho_1 \qquad \Gamma, \mathsf{z} : \langle\Gamma\rangle(\rho_1) \vdash_{\downarrow} \mathsf{t}_2 : \sigma_2}{\Gamma \vdash_{\downarrow} \mathsf{let}\ \mathsf{z} = \mathsf{t}_1\ \mathsf{in}\ \mathsf{t}_2 : \sigma_2}$$

Let-I

Fun-Ci

Fun-Ie

Fun-Ce

$$\frac{\Gamma, \mathsf{z} : \sigma[\bar\tau/\bar\beta] \vdash_{\downarrow} \mathsf{t} : \sigma_1 \qquad \sigma_2 \leq_p^{\shortparallel} \sigma[\bar\tau/\bar\beta] \qquad \bar\alpha \notin \mathsf{ftv}(\Gamma)}{\Gamma \vdash_{\downarrow} \mathsf{fun}\ (\mathsf{z} : \exists\,\beta.\,\sigma)\ \mathsf{t} : \forall\,\bar\alpha.\,\sigma_2 \to \sigma_1}$$

Fun-Ii

$$\frac{\Gamma, \mathsf{z} : \tau \vdash_{\uparrow} \mathsf{t} : \rho}{\Gamma \vdash_{\uparrow} \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau \to \rho}$$

Syntactic suggar:

$$(\mathsf{t} : \sigma) \quad = \quad (\mathsf{fun}\ (\mathsf{z} : \sigma)\ \mathsf{z})\ \mathsf{t}$$

Results

- Bidirectional shape inference is equivalent to PJS when restricted to canonical types
- We need not restrict to canonical types.
  - This is important for use with side effects.
  - Arguments of applications are checked, which avoids the weakness of shape inference.
  - Annotations can be used to change from inference to checking mode.
- Inference with value restriction is safe (because $\mathsf{F}_p^{\Downarrow}$ is safe).
- Completeness of inference the rules with value restriction reduces to completeness for $\mathsf{F}_p^{\Downarrow}$.
- However, in inference mode some power of OL is still missing. Is it bad? Do we have to mix shape and monotype inference to recover it?

Conclusions

Split shape inference and monotype inference

- Each one is easy to understand
- The core language $\mathsf{F}_p^{\Downarrow}$ is shared
  including completeness of monotype inference.
- The algorithmic aspect of the specification (bidirectional propagation) is simpler (need not think about unification)

Back into some well-understood framework

- F^{η} for soundness.
- Type-constraints for ML-like type inference.
- Closed types for shape inference.

Conclusions

Split shape inference and monotype inference

Back into some well-understood framework

Still more investigation needed for

- non canonical types.
- value restriction.

Conclusions

Split shape inference and monotype inference

Back into some well-understood framework

Still more investigation needed for non canonical types

Other applications of this framework

- More agressive shape inference?
  e.g. colored local type inference?
  propagate annotations by unification?
- extend the technique to higher-order F^ω?
- $F_\leq$ with explicit second-order subtyping and implicit first-order subtyping constraintes?
- Can this framework be used to simplify wobbly types?

Conclusions

Split shape inference and monotype inference

Back into some well-understood framework

Still more investigation needed for non canonical types

Other applications of this framework

- More agressive shape inference?
  e.g. colored local type inference?
  propagate annotations by unification?
- extend the technique to higher-order F^ω?
- $\mathsf{F}_{\leq}$ with explicit second-order subtyping and implicit first-order subtyping constraintes?
- Can this framework be used to simplify wobbly types?

Questions?

Thank you.

Algorithm $$(\varphi \wedge \Gamma \vdash \mathtt{t} : \rho) \rightsquigarrow \varphi'$$

Given a partial solution φ, and a type inference problem $\Gamma \vdash \mathtt{t} : \rho$, the algorithm computes a best solution φ'.

Thats is, φ' is the most general substitution that is both less general then φ and satisfies the typing problem (*i.e.* such that $\varphi'(\Gamma) \vdash \mathtt{t} : \varphi'(\rho)$). *Most general* means that other solutions are of the form $\varphi'' \circ \varphi'$.

In reality, we must keep track of fresh variables, and rather write $\exists W.(\varphi \wedge \Gamma \vdash \mathtt{t} : \rho) \rightsquigarrow \exists W'. \varphi'$. where W is the set of variables introduced by φ.

Algorithm

$$(\varphi \wedge \Gamma \vdash \mathsf{t} : \rho) \rightsquigarrow \varphi'$$

$$\frac{\varphi(\Gamma(\mathsf{z})) \leq \varphi(\sigma) \rightsquigarrow \varphi'}{\varphi \wedge \Gamma \vdash \mathsf{z} : \sigma \rightsquigarrow \varphi' \circ \varphi}$$

$$\frac{\begin{array}{c}\varphi \wedge \Gamma \vdash \mathsf{t}_1 : \sigma_2 \to \rho_1 \rightsquigarrow \varphi_1 \\ \varphi_1 \wedge \Gamma \vdash \mathsf{t}_2 : \sigma_2 \rightsquigarrow \varphi_2 \qquad \bar{\beta} \text{ fresh}\end{array}}{\varphi \wedge \Gamma \vdash \mathsf{t}_1\ (\mathsf{t}_2 : \exists\, \bar{\beta}.\, \sigma_2) : \rho_1 \rightsquigarrow \varphi_2}$$

$$\frac{\varphi \wedge \Gamma, \mathsf{z} : \sigma \vdash \mathsf{t} : \rho \rightsquigarrow \varphi'}{\varphi \wedge \Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \sigma \to \rho \rightsquigarrow \varphi'}$$

$$\frac{\begin{array}{c}\varphi(\tau) = \beta' \to \beta \rightsquigarrow \varphi' \qquad \beta\beta' \text{ fresh} \\ \varphi' \wedge \Gamma, \mathsf{z} : \beta' \vdash \mathsf{t} : \beta \rightsquigarrow \varphi''\end{array}}{\varphi \wedge \Gamma \vdash \mathsf{fun}\ (\mathsf{z})\ \mathsf{t} : \tau \rightsquigarrow \varphi''}$$

$$\frac{\begin{array}{c}\varphi \wedge \Gamma \vdash \mathsf{t}_1 : \rho_1 \rightsquigarrow \varphi_1 \qquad \bar{\beta} \text{ fresh} \\ \varphi_1 \wedge \Gamma, \mathsf{z} : \langle\Gamma\rangle(\varphi_1(\sigma_1)) \vdash \mathsf{t}_2 : \forall\, \bar{\alpha}.\, \rho_2 \rightsquigarrow \varphi_2\end{array}}{\varphi_2 \wedge \Gamma \vdash \mathsf{let}\ \mathsf{z} = (\mathsf{t}_1 : \exists\, \bar{\beta}.\, \sigma_1)\ \mathsf{in}\ \mathsf{t}_2 : \rho_1}$$

Algorithm

$$(\varphi \wedge \Gamma \vdash \mathtt{t} : \rho) \leadsto \varphi'$$

It can be (advantageously) seen as a type constraints with a particular eager resolution strategy.

Boxed Polymorphism

This is the poor man polymorphism: use a data-type constructor to automatically *encapsulate* a polymorphic type as an ML type and its associated destructor to *project* it back later into a polymorphic type.

$$\text{type } \mathit{id}\ \alpha = \mathit{Id} \text{ of } \forall\ \alpha.\ (\alpha \to \alpha)$$

Using the constructeur at both introduction and elimination points is then sufficient:

$$\text{let } \mathit{id} = \mathit{Id}\ (\text{fun } x \to x)$$

$$\text{let } \mathit{auto}\ (\mathit{Id}\ f) = f\ f$$

$$\mathit{auto}\ \mathit{id}$$

Boxed Polymorphism

This is the poor man polymorphism: use a data-type constructor to automatically *encapsulate* a polymorphic type as an ML type and its associated destructor to *project* it back later into a polymorphic type.

$$\text{type } \mathit{id}\ \alpha = \mathit{Id} \text{ of } \forall\, \alpha.\, (\alpha \to \alpha)$$

Limitations

- Simple cases are easy, but may become tricky when quantifiers are appear under other quantifiers, etc.
- An polymorphic type can often be embeded into an ML type several (incompatible) ways.
- Becomes heavy for an intensive usage:
  - type declaration is needed, even for a single use.
  - types are less readable—each (group of) quantifiers must be named.