

Generic pattern unification

A categorical approach

Ambroise Lafont Neel Krishnaswami

University of Cambridge

December 2022

A quick introduction to unification

$$\underbrace{t} \stackrel{?}{=} \underbrace{u}$$

terms with metavariables M, N, \dots

Unifier = metavariable substitution σ s.t.

$$t[\sigma] = u[\sigma]$$

Most general unifier = unifier σ that uniquely factors any other

$$\forall \delta, \quad t[\delta] = u[\delta] \quad \Leftrightarrow \quad \exists! \delta'. \quad \delta = \delta' \circ \sigma$$

Goal of unification = find the most general unifier

Where is unification used?

First-order unification

No metavariable argument

Examples

- Logic programming (Prolog)
- ML type inference systems

$$(M \rightarrow N) \stackrel{?}{=} (\mathbb{N} \rightarrow M)$$

Second-order unification

$M(\dots)$

Examples

- λ -Prolog
- Type theory, proof assistants

$$(\forall x.M(x, u)) \stackrel{?}{=} t$$

Undecidable

Pattern unification [Miller '91]

A **decidable** fragment of second-order unification.

Pattern restriction:

$$M(\underbrace{x_1, \dots, x_n}_{\text{distinct variables}})$$

\exists unification algorithm [Miller '91]

- fails if no unifier
- returns the most general unifier
- linear complexity [Qian '96]

This work

A **generic** algorithm for pattern unification

- Parameterised by a *signature*
- Categorical semantics

Examples

- *binding signatures*
- Linear syntax (e.g., quantum λ -calculus)
- Intrinsic system F

See our preprint.

Related work: algebraic accounts of unification

First-order unification

- Lattice theory [Plotkin '70]
- Category theory
 - [Rydeheard-Burstall '88]
 - [Goguen '89]

Pattern unification

- Category theory
 - [Vezzosi-Abel '14]
normalised λ -terms
 - [This work](#)

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Syntax (De Bruijn levels)

Metavariable context $(M_1 : m_1, \dots)$

$$\underbrace{\Gamma}_{\text{Variable context}} ; \underbrace{n}_{\text{Variable context}} \vdash t$$

$$\frac{1 \leq i \leq n}{\Gamma ; n \vdash v_i} \text{VAR}$$

$$\frac{\Gamma ; n \vdash t \quad \Gamma ; n \vdash u}{\Gamma ; n \vdash t u} \text{APP}$$

$$\frac{\Gamma ; n + 1 \vdash t}{\Gamma ; n \vdash \lambda t} \text{ABS}$$

$$\frac{(M : m) \in \Gamma \quad 1 \leq i_1, \dots, i_m \leq n \quad i_1, \dots, i_m \text{ distinct}}{\Gamma ; n \vdash M(v_{i_1}, \dots, v_{i_m})} \text{FLEX}$$

No β/η -equation.

Metavariable substitution

Substitution σ from $\overbrace{(M_1 : m_1, \dots, M_p : m_p)}^{\Gamma}$ to Δ :

$$(\sigma_1, \dots, \sigma_p) \quad \text{s.t.} \quad \Delta; m_i \vdash \sigma_i$$

Notation

$$M_i(v_1, \dots, v_{m_i}) \mapsto \sigma_i$$

Term substitution

$$\Gamma; n \vdash t \quad \mapsto \quad \Delta; n \vdash t[\sigma]$$

Base case:

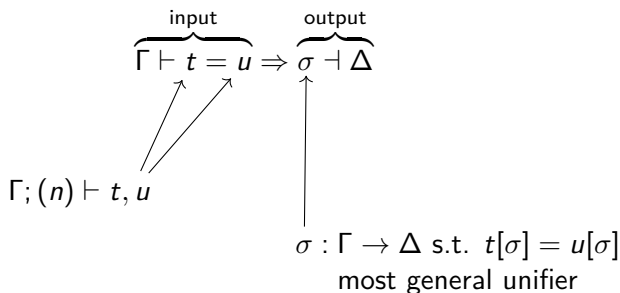
$$M_i(x_1, \dots, x_{m_i}) \mapsto \sigma_i[v_j \mapsto x_j]$$

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Unification algorithm

make the n explicit?



Examples

$$\Gamma, M : 2 \vdash M(x, y) = x \Rightarrow (M(v_1, v_2) \mapsto v_1) \dashv \Gamma$$

$$\Gamma, M : 2 \vdash M(x, y) = y \Rightarrow (M(v_1, v_2) \mapsto v_2) \dashv \Gamma$$

Impossible cases

$$\Gamma \vdash \lambda t = u_1 u_2 \Rightarrow ! \vdash \perp$$

formal "error" context

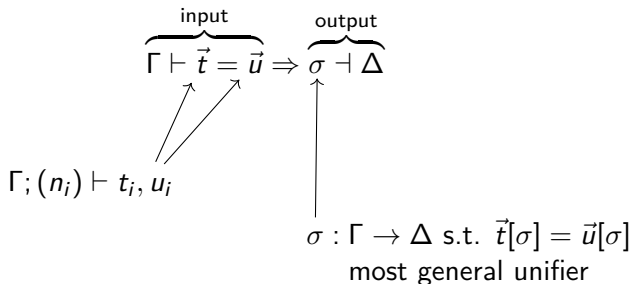
formal "error" substitution

Congruence

$$\frac{\Gamma \vdash t = u \Rightarrow \sigma \vdash \Delta}{\Gamma \vdash \lambda t = \lambda u \Rightarrow \sigma \vdash \Delta}$$

$$\frac{\Gamma \vdash "t_1, t_2 = u_1, u_2" \Rightarrow \sigma \vdash \Delta}{\Gamma \vdash t_1 \ t_2 = u_1 \ u_2 \Rightarrow \sigma \vdash \Delta}$$

Unifying lists of terms



Sequential unification

$$\frac{\Gamma \vdash t_1 = u_1 \Rightarrow \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash \vec{t}_2[\sigma_1] = \vec{u}_2[\sigma_1] \Rightarrow \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t}_2 = u_1, \vec{u}_2 \Rightarrow \sigma_1[\sigma_2] \dashv \Delta_2} \text{U-SPLIT}$$

Unifying a metavariable $M(\vec{x}) \stackrel{?}{=} \dots$

Three cases

- 1 $M(\vec{x}) \stackrel{?}{=} M(\vec{y})$
- 2 $M(\vec{x}) \stackrel{?}{=} \dots M(\vec{y}) \dots$
- 3 $M(\vec{x}) \stackrel{?}{=} u$ and $M \notin u$ (non-cyclic)

Unifying a metavariable with itself

$$M(x_1, \dots, x_m) \stackrel{?}{=} M(y_1, \dots, y_m)$$

Most general unifier

\vec{p} = vector of common positions: $(x_{p_1}, \dots, x_{p_n}) = (y_{p_1}, \dots, y_{p_n})$

$$\sigma : M(v_1, \dots, v_m) \mapsto N(v_{p_1}, \dots, v_{p_n})$$

Examples

$$\begin{array}{c} \vec{p} = (2) \\ \overbrace{M(x, y) = M(z, y)} \Rightarrow M(v_1, v_2) \mapsto N(v_2) \\ \overbrace{M(x, y) = M(z, x)} \Rightarrow M(v_1, v_2) \mapsto N \\ \vec{p} = () \end{array}$$

Deep cyclic case

$$M(\vec{x}) \stackrel{?}{=} \dots M(\vec{y}) \dots$$

No unifier

$$\underbrace{M(\vec{x})} = \underbrace{\dots M(\vec{y}) \dots} \Rightarrow ! \vdash \perp$$

sizes cannot match after substitution

Non-cyclic case

$$M(\vec{x}) \stackrel{?}{=} u \text{ (} M \notin u \text{)} \quad (1)$$

Most general unifier

(1) as the definition of M :

$$\sigma : M(v_1, \dots, v_m) \mapsto u[x_i \mapsto v_i]$$

Side condition

$$fv(u) \subset \vec{x}$$

$M(x) \stackrel{?}{=} y$ has no unifier ($x \neq y$)

Pruning

What about $M(x) \stackrel{?}{=} \underbrace{N(x, y)}_u$?

Most general unifier

$$N(v_1, v_2) \mapsto M(v_1)$$

- Side-condition $fv(u) \subset \vec{x}$ is too pessimistic.
- Can be enforced by restricting metavariable arities in u .

$$N(x, y) \xrightarrow{\text{pruning}} N'(x)$$

Non-cyclic phase

$$\Gamma \vdash u :> M(\vec{x}) \Rightarrow w; \sigma \vdash \Delta$$

$\Gamma; (n) \vdash u$ $M \notin \Gamma$ $n \vdash \vec{x}$

$\sigma : \Gamma \rightarrow \Delta$
 pruning substitution

Formal meaning

$$(\sigma, M(v_1, \dots, v_m) \mapsto w) = \text{most general unifier for } M(\vec{x}) \stackrel{?}{=} u$$

Pruning a variable

$$\frac{y \notin \vec{x}}{\Gamma \vdash y :> M(\vec{x}) \Rightarrow !; ! \dashv \perp} \text{VAR-FAIL}$$

$$\overline{\Gamma \vdash x_n :> M(\vec{x}) \Rightarrow v_n; id_{\Gamma} \dashv \Gamma}$$

Pruning a metavariable

$$M(\vec{x}) \stackrel{?}{=} N(\vec{y}) \quad (M \neq N)$$

Most general unifier

\vec{l}, \vec{r} = vectors of common value positions:

$$(x_{l_1}, \dots, x_{l_p}) = (y_{r_1}, \dots, y_{r_p})$$

Then,

$$M(v_1, \dots, v_m) \mapsto P(v_{l_1}, \dots, v_{l_p})$$

$$N(v_1, \dots, v_n) \mapsto P(v_{r_1}, \dots, v_{r_p})$$

Examples

Intuition?

$$M(x, y) = N(z, x) \Rightarrow M(v_1, v_2) \mapsto P(v_1)$$

$$N(v_1, v_2) \mapsto P(v_2)$$

$$M(x, y) = N(z) \Rightarrow M(v_1, v_2) \mapsto P, N(v_1) \mapsto P$$

Pruning operations

Divide & Conquer: a fresh metavariable for each argument.

$$\frac{\Gamma \vdash t :> M'(\vec{x}, \overbrace{v_{n+1}}^{\text{bound variable}}) \Rightarrow w; \sigma \vdash \Delta}{\Gamma \vdash \lambda t :> M(\vec{x}) \Rightarrow \lambda w; \sigma \vdash \Delta} \quad M = \lambda M'$$

$$\frac{\text{"}\Gamma \vdash t, u :> M_1(\vec{x}), M_2(\vec{x}) \Rightarrow w_1, w_2; \sigma \vdash \Delta\text{"}}{\Gamma \vdash t \ u :> M(\vec{x}) \Rightarrow w_1 \ w_2; \sigma \vdash \Delta} \quad M = M_1 \ M_2$$

Non-cyclic unification of multi-terms

$$\Gamma \vdash u_1, \dots, u_n :> M_1(\vec{x}_1), \dots, M_n(\vec{x}_n) \Rightarrow w_1, \dots, w_n; \sigma \vdash \Delta$$

$\Gamma; (n_i) \vdash u_i$
 $(M_i : m_i) \notin \Gamma$
 $\Delta; m_i \vdash w_i$
 $\sigma : \Gamma \rightarrow \Delta$

Formal meaning

$(\sigma, M_i(v_1, \dots, v_{m_i}) \mapsto w_i) = \text{most general unifier for}$

$$M_1(\vec{x}_1), \dots, M_n(\vec{x}_n) \stackrel{?}{=} u_1, \dots, u_n$$

Sequential non-cyclic unification

$$\frac{\Gamma \vdash t_1 :> M_1(\vec{x}) \Rightarrow u_1; \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash \vec{t}_2[\sigma_1] :> \vec{M}_2 \Rightarrow \vec{u}_2; \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t}_2 :> M_1(\vec{x}), \vec{M}_2 \Rightarrow u_1[\sigma_2], \vec{u}_2; \sigma_1[\sigma_2] \dashv \Delta_2}$$

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Parameterisation by a *signature*

Binding signature for pure λ -calculus

$app : (0, 0)$ $abs : (1)$

number of bound variables in the argument

Example: $M(\vec{x}) \stackrel{?}{=} o(\vec{t})$, non-cyclic

$$o : (\alpha_1, \dots, \alpha_p)$$

$$\frac{\Gamma \vdash \vec{t} :> M_1(\vec{x}, \overbrace{v_{n+1}, \dots, v_{n+1+\alpha_1}}^{\text{bound variables}}), \dots, M_p(\dots) \Rightarrow \vec{u}; \sigma \vdash \Delta}{\Gamma \vdash o(\vec{t}) :> M(\vec{x}) \Rightarrow o(\vec{u}); \sigma \vdash \Delta}$$

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 **Categorical generalisation**
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Pure λ -calculus as a functor

category of finite cardinals and injections between them



Pure λ -calculus as a functor $\Lambda : \mathbb{F}_m \rightarrow \mathbf{Set}$

$$\Lambda_n = \{t \mid \cdot; n \vdash t\}$$

injective renaming

$$M(\vec{x})[\sigma] = \sigma_M \overbrace{[v_i \mapsto x_i]}$$

Pure λ -calculus as a fixpoint

$$\Lambda_n \cong \underbrace{\{1, \dots, n\}}_{\text{variables}} + \underbrace{\Lambda_n \times \Lambda_n}_{\text{application}} + \underbrace{\Lambda_{n+1}}_{\text{abstraction}}$$

In fact,

$$\Lambda = \mu X. F(X)$$

Initial algebra of the endofunctor F on $[\mathbb{F}_m, \text{Set}]$

$$F(X)_n = \{0, \dots, n-1\} + X_n \times X_n + X_{n+1}$$

Pure λ -calculus extended with a metavariable $M : m$

$$\Lambda_n^{M:m} = \{t \mid M : m; n \vdash t\}$$

As an initial algebra:

$$\Lambda^{M:m} = \mu X. \underbrace{(F(X))}_{\text{operations / variables}} + \underbrace{\text{metavariables}}_{arg^M}$$

$$= \underbrace{T}_{\text{free monad generated by } F}(arg^M)$$

Pure λ -calculus extended with a metavariable $M : m$

$$\text{arg}^M : \mathbb{F}_m \rightarrow \text{Set}$$

$$\begin{aligned} \text{arg}^M_n &= \{M\text{-arguments in the variable context } \mathbf{n}\} \\ &= \{\text{choice of } m \text{ distinct variables in the context } \mathbf{n}\} \\ &= \text{Inj}(m, n) \\ &= \text{hom}_{\mathbb{F}_m}(m, n) = ym_n \end{aligned}$$

$$\Lambda^{M:m} = T(ym)$$

Pure λ -calculus with metavariables

Given a metavariable context Γ , define

$$\underline{\Gamma} := \coprod_{(M:m) \in \Gamma} ym$$

$$T(\underline{\Gamma})_n = \{t \mid \Gamma; n \vdash t\}$$

Unification as a Kleisli coequaliser

Claims¹:

- $\text{hom}(yn, T\underline{\Gamma}) = \text{set of terms in context } \Gamma; n.$
- $\text{hom}(\underline{\Gamma}, T\underline{\Delta}) = \text{set of metavariable substitutions } \Gamma \rightarrow \Delta.$
- Most general unifier of t, u : coequaliser of $yn \xRightarrow[t]{u} T\underline{\Gamma}$
in $\text{Th}^{op}(F) \subset \text{Kl}(T).$

Objects: $\underline{\Gamma}, \underline{\Delta}, \dots$

"Lawvere theory"

¹well-known in the first-order case.

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation**
 - A case study: syntax of pure λ -calculus
 - **Generic pattern unification**
- 4 Example: System F

Signature

- 1 \mathcal{A} small category of contexts (e.g., $\mathbb{F}_m, 1$)

Intuition: objects = metavariable arities,
morphisms = metavariable arguments.

- All morphisms in \mathcal{A} are monic (pattern restriction).
- \mathcal{A} has equalisers and pullbacks ($M(\vec{x}) \stackrel{?}{=} N(\vec{y})$).

- 2 F endofunctor on $[\mathcal{A}, \text{Set}]$ of the shape

$$F(X)_a = \coprod_{o \in O_a} X_{L_{o,1}} \times \cdots \times X_{L_{o,n_o}}$$

such that F restricts to an endofunctor on functors preserving finite connected limits.

Typing rules

Notation

$$\underbrace{\Gamma; b \vdash u}_{M_1 : a_1, \dots, M_n : a_n} \quad \text{means} \quad u \in T(\underbrace{\Gamma}_{ya_1 + \dots + ya_n})_b$$

$$F(X)_a = \coprod_{o \in O_a} X_{L_{o,1}} \times \dots \times X_{L_{o,n_o}}$$

$$\frac{\Gamma; L_{o,i} \vdash t_i}{\Gamma; a \vdash o(\vec{t})} \text{RIGID} \qquad \frac{x \in \text{hom}_{\mathcal{A}}(a, b)}{\Gamma, M : a \quad ; \quad b \vdash M(x)} \text{FLEX}$$

No equation.

Semantics of unification

Claim: Given a signature (\mathcal{A}, F) , a coequaliser diagram in $\text{Th}^{op}(F)$ has a colimit as soon as there exists a cocone (i.e., a ‘unifier’).

Proof: By describing a unification algorithm.

End of this section: soundness proofs for 3 rules.

Interpreting the unification statements

Notations

$$\Gamma \vdash t = u \Rightarrow \sigma \dashv \Delta \Leftrightarrow \cdot \begin{array}{c} \xrightarrow{t} \\ \xleftarrow{u} \end{array} \Gamma - \frac{\sigma}{\cdot} \gg \Delta \quad \text{coequaliser}$$

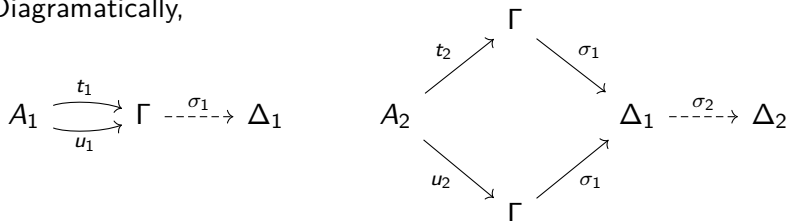
$$\Gamma \vdash t :> f \Rightarrow v; \sigma \dashv \Delta \Leftrightarrow \begin{array}{ccc} \cdot & \xrightarrow{f} & \cdot \\ t \downarrow & & \downarrow v \\ \Gamma - \frac{\sigma}{\cdot} & \gg & \Delta \end{array} \quad \text{pushout}$$

mostly used in $\text{Th}^{op}(F)_{\perp} = \text{Th}^{op}(F) + \text{a free terminal object } \perp$.

Soundness of U-SPLIT [Rydeheard-Burstall '88]

$$\frac{\Gamma \vdash t_1 = u_1 \Rightarrow \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash t_2[\sigma_1] = u_2[\sigma_1] \Rightarrow \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, t_2 = u_1, u_2 \Rightarrow \sigma_1[\sigma_2] \dashv \Delta_2} \text{U-SPLIT}$$

Diagrammatically,



$$A_1 + A_2 \xRightarrow[u_1, u_2]{t_1, t_2} \Gamma \xRightarrow{\sigma_2 \circ \sigma_1} \Delta_2$$

Soundness of U-FLEXFLEX

$$\frac{b \vdash x =_{\mathcal{A}^{op}} y \Rightarrow z \vdash c}{M : b \vdash M(x) = M(y) \Rightarrow M \mapsto M'(z) \vdash M' : c} \text{U-FLEXFLEX}$$

Diagrammatically, faire le line explicite avec le pure lamdba

$$\frac{a \xrightarrow[x]{y} b - \frac{z}{\Rightarrow} c \quad \text{in } \mathcal{A}^{op}}{\mathcal{L}a \xrightarrow[\mathcal{L}y]{\mathcal{L}x} \mathcal{L}b - \frac{\mathcal{L}z}{\Rightarrow} \mathcal{L}c \quad \text{in } \text{Th}^{op}(F)}$$

where

$$a \xrightarrow{x} b \quad \xrightarrow{\mathcal{L} : \mathcal{A}^{op} \rightarrow \text{Th}^{op}(F)} \quad ya \xrightarrow{"M(x)"} T(\underline{M : b})$$

Soundness of U-NoCYCLE

$$\frac{u|_{\Gamma} = \underline{u'} \quad \Gamma \vdash u' :> M(x) \Rightarrow v; \sigma \vdash \Delta}{\Gamma, M : b \vdash M(x) = u \Rightarrow \sigma, M \mapsto v \vdash \Delta} \text{U-NoCYCLE}$$

Diagrammatically,

$$\begin{array}{ccc} ya & \xrightarrow{"M(x)"} & yb \\ u' \downarrow & & \downarrow v \\ \Gamma - \frac{}{\sigma} & \gg & \Delta \end{array} \quad \text{pushout}$$

$$\begin{array}{ccccc} & & yb & & \\ & \nearrow^{"M(x)"} & & \searrow_{in_2} & \\ ya & & & & \Gamma + yb \begin{smallmatrix} [v, \sigma] \\ - - \end{smallmatrix} \gg \Delta \\ & \searrow_{u'} & & \nearrow_{in_1} & \\ & & \Gamma & & \end{array} \quad \text{coequaliser}$$

Outline

- 1 Pattern unification for pure λ -calculus
 - Syntax
 - Unification algorithm
- 2 Generalisation to binding signatures
- 3 Categorical generalisation
 - A case study: syntax of pure λ -calculus
 - Generic pattern unification
- 4 Example: System F

Types

Notation

$n \vdash \tau$ type \Leftrightarrow the type τ is wellformed in context n

$$\frac{1 \leq i \leq n}{n \vdash \eta_i \text{ type}} \text{TYPE-VAR} \qquad \frac{n+1 \vdash \tau \text{ type}}{n \vdash \forall \tau \text{ type}} \text{FORALL}$$

$$\frac{n \vdash \tau_1, \tau_2 \text{ type}}{n \vdash \tau_1 \rightarrow \tau_2 \text{ type}} \text{ARROW}$$

Metavariable arities

Metavariable application

$$M(\overbrace{\alpha_1, \dots, \alpha_p}^{\text{type variables}} \mid \overbrace{x_1, \dots, x_q}^{\text{"ground" variables}})$$

$$M : (p \mid \overbrace{\tau_1, \dots, \tau_q}^{\substack{p \vdash \tau_i \text{ type} \\ \text{input argument types}}} \vdash \tau_f)$$

number of type variable arguments

output type

Signature

- Objects of \mathcal{A} = metavariables arities

$$n|\vec{\tau} \vdash \sigma_f$$

- Endofunctor F on $[\mathcal{A}, \text{Set}]$ s.t.

$$\mu F(n|\vec{\tau} \vdash \sigma_f) = \{t \text{ s.t. } n|\vec{\tau} \vdash t : \sigma_f\}$$

The endofunctor for System F

Typing rule	$F(X)_{n \vec{\tau} \vdash \sigma_f} = \coprod \dots$
$\frac{\tau_i = \sigma_f}{n \vec{\tau} \vdash v_i : \sigma_f} \text{VAR}$	$ \vec{\tau} _{\sigma_f}$
$\frac{n \vec{\tau}, \sigma_1 \vdash t : \sigma_2}{n \vec{\tau} \vdash \lambda t : \sigma_1 \rightarrow \sigma_2} \text{ABS}$	$\coprod_{\sigma_1, \sigma_2 \text{ s.t. } \sigma_f = (\sigma_1 \rightarrow \sigma_2)} X_{n \vec{\tau}, \sigma_1 \vdash \sigma_2}$
$\frac{n \vec{\tau} \vdash t : \sigma \rightarrow \sigma_f \quad n \vec{\tau} \vdash u : \sigma_f}{n \vec{\tau} \vdash t u : \sigma_f} \text{APP}$	$\coprod_{\sigma} X_{n \vec{\tau} \vdash \sigma \rightarrow \sigma_f} \times X_{n \vec{\tau} \vdash \sigma}$
$\frac{n+1 \vec{\tau} \vdash t : \sigma}{n \vec{\tau} \vdash \Lambda t : \forall \sigma} \text{T-ABS}$	$\coprod_{\sigma \text{ s.t. } \sigma_f = \forall \sigma} X_{n+1 \vec{\tau} \vdash \sigma}$
$\frac{n \vec{\tau} \vdash t : \forall \sigma_1}{n \vec{\tau} \vdash t \cdot \sigma_2 : \sigma_1[\sigma_2]} \text{T-APP}$	$\coprod_{\sigma_1, \sigma_2 \text{ s.t. } \sigma_f = \sigma_1[\sigma_2]} X_{n \vec{\tau} \vdash \forall \sigma_1}$

Unification in system F: an example

$$M(\vec{\alpha}|\vec{x}) \stackrel{?}{=} M(\vec{\beta}|\vec{y})$$

Most general unifier

$$M(\eta_1, \dots, \eta_p | v_1, \dots, v_m) \mapsto N(\vec{\gamma} | \vec{z})$$

where $\vec{\gamma}$ and \vec{z} are vectors of common positions

$$\alpha_{\vec{\gamma}} = \beta_{\vec{\gamma}}$$

$$x_{\vec{z}} = y_{\vec{z}}$$

Summary of the generic unification algorithm

$$\overline{\Gamma \vdash () = () \Rightarrow id_{\Gamma} \dashv \Gamma} \quad \overline{\perp \vdash \vec{t} = \vec{u} \Rightarrow ! \dashv \perp}$$

$$\frac{\Gamma \vdash t_1 = u_1 \Rightarrow \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash \vec{t}_2[\sigma_1] = \vec{u}_2[\sigma_1] \Rightarrow \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t}_2 = u_1, \vec{u}_2 \Rightarrow \sigma_1[\sigma_2] \dashv \Delta_2} \text{U-SPLIT}$$

$$\frac{\Gamma \vdash \vec{t} = \vec{u} \Rightarrow \sigma \dashv \Delta}{\Gamma \vdash o(\vec{t}) = o(\vec{u}) \Rightarrow \sigma \dashv \Delta} \text{U-RIGRIG} \quad \frac{o \neq o'}{\Gamma \vdash o(\vec{t}) = o'(\vec{u}) \Rightarrow ! \dashv \perp}$$

$$\frac{u|_{\Gamma} = u' \quad \Gamma \vdash u' :> M(x) \Rightarrow v; \sigma \dashv \Delta}{\Gamma, M : b \vdash M(x) = u \Rightarrow \sigma, M \mapsto v \dashv \Delta} \text{U-NOCYCLE} + \text{sym}$$

$$\frac{b \vdash x =_{\mathcal{A}^{op}} y \Rightarrow z \dashv c}{\Gamma, M : b \vdash M(x) = M(y) \Rightarrow M \mapsto M'(z) \dashv \Gamma, M' : c} \text{U-FLEXFLEX}$$

$$\frac{u = o(\vec{t}) \quad u|_{\Gamma} \neq \dots}{\Gamma, M : b \vdash M(x) = u \Rightarrow ! \dashv \perp} \text{U-CYCLIC} + \text{sym}$$

Non cyclic phase

$$\overline{\Gamma \vdash () :> () \Rightarrow (); id_{\Gamma} \dashv \Gamma} \quad \overline{\perp \vdash \vec{t} :> \vec{f} \Rightarrow !; ! \dashv \perp}$$

$$\frac{\Gamma \vdash t_1 :> M_1(\vec{x}) \Rightarrow u_1; \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash \vec{t}_2[\sigma_1] :> \vec{M}_2 \Rightarrow \vec{u}_2; \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t}_2 :> M_1(\vec{x}), \vec{M}_2 \Rightarrow u_1[\sigma_2], \vec{u}_2; \sigma_1[\sigma_2] \dashv \Delta_2}$$

$$\frac{c \vdash_{\mathcal{A}^{op}} y :> x \Rightarrow l; r \dashv d}{\Gamma, M : c \vdash M(y) :> N(x) \Rightarrow M'(l); M \mapsto M'(r) \dashv \Gamma, M' : d} \text{P-FLEX}$$

$$\frac{\Gamma \vdash \vec{t} :> \mathcal{L}^+ x^o \Rightarrow \vec{u}; \sigma \dashv \Delta \quad o = x \cdot o'}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow o'(\vec{u}); \sigma \dashv \Delta}$$

$$\frac{o \neq x \cdot \dots}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow !; ! \dashv \perp}$$

Pruning $o(\vec{t})$: variable case

o is a variable $\Rightarrow \vec{t} = ()$.

$$\begin{array}{c}
 \frac{\Gamma \vdash \overbrace{\vec{t}}^{()} :> \dots \quad \overbrace{o = x_{o'}}^{o = \vec{x} \cdot o'}}{\Gamma \vdash o(\vec{t}) :> N(\vec{x}) \Rightarrow o'(\vec{u}); \sigma \vdash \Delta} \Leftrightarrow \frac{o = x_i}{\Gamma \vdash o :> N(\vec{x}) \Rightarrow i; id_{\Gamma} \vdash \Gamma} \\
 \\
 \frac{o \neq x \cdot \dots}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow !; ! \vdash \perp} \Leftrightarrow \frac{o \notin \vec{x}}{\Gamma \vdash o :> N(\vec{x}) \Rightarrow !; ! \vdash \perp}
 \end{array}$$

Pruning $o(\vec{t})$: operation case

Assume $o : (\alpha_1, \dots, \alpha_p)$

$$\frac{\Gamma \vdash \vec{t} :> \overbrace{\mathcal{L}^+ x^o}^{M_1(\vec{x}, n, \dots, n+\alpha_1-1), \dots, M_p(\dots)} \Rightarrow \vec{u}; \sigma \vdash \Delta \quad \overbrace{o = o'}^{o = x \cdot o'}}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow o'(\vec{u}); \sigma \vdash \Delta}$$

$$\frac{o \neq x \cdot \dots}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow !; ! \vdash \perp} \quad \text{never applies}$$

Typing rule for metavariables

Typing judgement

$$\underbrace{\Gamma}_{\text{Metavariable context}} ; \underbrace{n}_{\text{Type variable context}} \mid \underbrace{t_1, \dots, t_m \vdash u : t_f}_{\substack{\text{Types of variables} \\ n \vdash t_i \text{ type}}}$$

Typing metavariables

$$\frac{0 < \overbrace{\alpha_1, \dots, \alpha_p}^{\text{distinct}} \leq n \quad 0 < \overbrace{x_1, \dots, x_q}^{\text{distinct}} \leq m \quad \tau_i[\vec{\alpha}] = t_{x_i}}{\Gamma, M : (p \mid \tau_1, \dots, \tau_q \vdash \tau_f) ; n \mid t_1, \dots, t_m \vdash M(\underbrace{\vec{\alpha}}_{\text{type variables}} \mid \vec{x}) : \tau_f[\vec{\alpha}]}$$

Future directions

- Implementation
 - logic programming
 - verified in Coq/Agda
(needs rephrasing using structural recursion)
- Unification modulo reduction
- Efficient pattern unification [Qian '96]
- Anti-unification