# Generic pattern unification

## A categorical approach

Ambroise Lafont Neel Krishnaswami

University of Cambridge

December 2022

# A quick introduction to unification

$$\underbrace{t}_{} \overset{?}{=} \underbrace{u}_{}$$

terms with metavariables $M, N, \ldots$

**Unifier** $=$ metavariable substitution $\sigma$ s.t.

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

**Most general unifier** $=$ unifier $\sigma$ that uniquely factors any other

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

**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

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

**Second-order unification**

$M(\dots)$

**Examples**

- $\lambda$-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**:

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

$\exists$ 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 $\lambda$-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 $\lambda$-terms
  - This work

# Outline

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

# Outline

# Outline

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

# Syntax (De Bruijn levels)

$$\overbrace{\Gamma}^{\text{Metavariable context } (M_1 : m_1, \ldots)} ; \underbrace{n}_{\text{Variable context}} \vdash t$$

$$\frac{1 \leq i \leq n}{\Gamma ; n \vdash v_i}\textsc{Var} \qquad \frac{\Gamma ; n \vdash t \quad \Gamma ; n \vdash u}{\Gamma ; n \vdash t\ u}\textsc{App} \qquad \frac{\Gamma ; n+1 \vdash t}{\Gamma ; n \vdash \lambda t}\textsc{Abs}$$

$$\frac{(M : m) \in \Gamma \qquad 1 \leq i_1, \ldots, i_m \leq n \qquad i_1, \ldots i_m \text{ distinct}}{\Gamma ; n \vdash M(v_{i_1}, \ldots, v_{i_m})}\textsc{Flex}$$

No $\beta/\eta$-equation

# Metavariable substitution

**Substitution** $\sigma$ **from** $\overbrace{(M_1 : m_1, \dots, M_p : m_p)}^{\Gamma}$ **to** $\Delta$:

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

## 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 $\lambda$-calculus
   - Syntax
   - Unification algorithm
2. Generalisation to binding signatures
3. Categorical generalisation
   - A case study: syntax of pure $\lambda$-calculus
   - Generic pattern unification
4. Example: System F

# Unification algorithm

$$\overbrace{\Gamma \vdash t = u}^{\text{input}} \Rightarrow \overbrace{\sigma \dashv \Delta}^{\text{output}}$$

$\Gamma; (n) \vdash t, u$

$\sigma : \Gamma \to \Delta$ s.t. $t[\sigma] = u[\sigma]$
most general unifier

# 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

$$\Gamma \vdash \lambda t = u_1 \ u_2 \Rightarrow \; ! \dashv \bot$$

formal "error" context

formal "error" substitution

# Congruence

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

$$\frac{\Gamma \vdash \text{"}t_1, t_2 = u_{1,} u_2\text{"} \Rightarrow \sigma \dashv \Delta}{\Gamma \vdash t_1 \ t_2 = u_1 \ u_2 \Rightarrow \sigma \dashv \Delta}$$

# Unifying lists of terms

$$\overbrace{\Gamma \vdash \vec{t} = \vec{u}}^{\text{input}} \Rightarrow \overbrace{\sigma \dashv \Delta}^{\text{output}}$$

$\Gamma; (n_i) \vdash t_i, u_i$

$\sigma : \Gamma \to \Delta$ s.t. $\vec{t}[\sigma] = \vec{u}[\sigma]$
most general unifier

# Sequential unification

$$\frac{\Gamma \vdash t_1 = u_1 \Rightarrow \sigma_1 \dashv \Delta_1 \qquad \Delta_1 \vdash \vec{t_2}[\sigma_1] = \vec{u_2}[\sigma_1] \Rightarrow \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t_2} = u_1, \vec{u_2} \Rightarrow \sigma_1[\sigma_2] \dashv \Delta_2}\textsc{U-Split}$$

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

Three cases

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

# Unifying a metavariable with itself

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

## Most general unifier

$\vec{\mathrm{p}}$ = vector of common positions: $(x_{\mathrm{p}_1}, \ldots, x_{\mathrm{p}_n}) = (y_{\mathrm{p}_1}, \ldots, y_{\mathrm{p}_n})$

$$\sigma : M(v_1, \ldots, v_m) \mapsto N(v_{\mathrm{p}_1}, \ldots v_{\mathrm{p}_n})$$

## Examples

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

$$\underbrace{M(x, y) = M(z, x)}_{\vec{p} = ()} \Rightarrow M(v_1, v_2) \mapsto N$$

# Deep cyclic case

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

No unifier

$$\underbrace{M(\vec{x})} = \underbrace{\ldots M(\vec{y}) \ldots} \Rightarrow ! \dashv \perp$$

sizes cannot match after substitution

# Non-cyclic case

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

## Most general unifier

(1) as the definition of $M$:

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

## Side condition

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

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

# Pruning

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

## 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

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

$\Gamma; (n) \vdash u$

$M \notin \Gamma \qquad n \vdash \vec{x}$

$\sigma : \Gamma \to \Delta$

pruning substitution

## Formal meaning

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

# Pruning a variable

$$\frac{y \notin \vec{x}}{\Gamma \vdash y :> M(\vec{x}) \Rightarrow !; ! \dashv \bot}\textsc{Var-Fail}$$

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

# Pruning a metavariable

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

## 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

$$\begin{array}{lcr} M(x,y) = N(z,x) & \Rightarrow & M(v_1, v_2) \mapsto P(v_1) \\ & & N(v_1, v_2) \mapsto P(v_2) \\ M(x,y) = N(z) & \Rightarrow & M(v_1, v_2), N(v_1) \mapsto P \end{array}$$

# Pruning operations

**Divide & Conquer**: a fresh metavariable for each argument.

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

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

# Non-cyclic unification of multi-terms

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

$\Gamma; (n_i) \vdash u_i$ $\quad$ $\Delta; m_i \vdash w_i$

$(M_i : m_i) \notin \Gamma$ $\quad$ $\sigma : \Gamma \to \Delta$

## Formal meaning

$(\sigma, M_i(v_1, \dots, v_{m_i}) \mapsto w_i) =$ 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

# Parameterisation by a *signature*

## Binding signature for pure $\lambda$-calculus

$$\mathit{app} : (0, 0) \qquad \mathit{abs} : (1)$$

number of bound variables in the argument

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

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

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

# Outline

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

# Outline

# Pure $\lambda$-calculus as a functor

category of finite cardinals and injections between them

## Pure $\lambda$-calculus as a functor $\Lambda : \mathbb{F}_m \to \mathrm{Set}$

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

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

# 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 $[\mathbb{F}_m, \mathrm{Set}]$

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

# Pure $\lambda$-calculus extended with a metavariable $M : m$

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

As an initial algebra:

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

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

# Pure $\lambda$-calculus extended with a metavariable $M : m$

$$
\begin{aligned}
arg^M &: \mathbb{F}_m \to \mathrm{Set} \\
arg^M{}_{\boldsymbol{n}} &= \{M\text{-arguments in the variable context } \boldsymbol{n}\} \\
&= \{\text{choice of } m \text{ distinct variables in the context } \boldsymbol{n}\} \\
&= \mathrm{Inj}(m, n) \\
&= \hom_{\mathbb{F}_m}(m, n) = ym_n
\end{aligned}
$$

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

# Pure $\lambda$-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

**Claims**[1]:

- $\hom(yn, T\underline{\Gamma})$ = set of terms in context $\Gamma; n$.
- $\hom(\underline{\Gamma}, T\underline{\Delta})$ = set of metavariable substitutions $\Gamma \to \Delta$.
- Most general unifier of $t, u$: coequaliser of $yn \overset{t}{\underset{u}{\rightrightarrows}} T\underline{\Gamma}$ in $\mathrm{Th}^{op}(F) \subset \mathrm{Kl}(T)$.

Objects: $\underline{\Gamma}, \underline{\Delta}, \ldots$
(finite coproducts of representable functors)
"Lawvere theory"

[1] well-known in the first-order case.

# Outline

## 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 $[\mathcal{A}, \mathrm{Set}]$ 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

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

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

$$\frac{\Gamma; L_{o,i} \vdash t_i}{\Gamma; a \vdash o(\vec{t})}\textsc{Rigid} \qquad \frac{x \in \hom_{\mathcal{A}}(a, b)}{\Gamma, M : a \quad ; \quad b \vdash M(x)}\textsc{Flex}$$

# Semantics of unification

**Claim**: Given a signature $(\mathcal{A}, F)$, a coequaliser diagram in $\mathrm{Th}^{op}(F)$ 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

$\Gamma \vdash t = u \Rightarrow \sigma \dashv \Delta \quad \Leftrightarrow \quad$ $\begin{tikzcd} \cdot \arrow[r, shift left, "t"] \arrow[r, shift right, "u"'] & \Gamma \arrow[r, dashed, "\sigma"] & \Delta \end{tikzcd}$ coequaliser

$\Gamma \vdash t :> f \Rightarrow v; \sigma \dashv \Delta \quad \Leftrightarrow \quad$ $\begin{tikzcd} \cdot \arrow[r, "f"] \arrow[d, "t"'] & \cdot \arrow[d, dashed, "v"] \\ \Gamma \arrow[r, dashed, "\sigma"'] & \Delta \end{tikzcd}$ pushout

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

# Soundness of U-Split [Rydeheard-Burstall '88]

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

Diagramatically,

$$\begin{tikzcd} A_1 \arrow[r, shift left, "t_1"] \arrow[r, shift right, "u_1"'] & \Gamma \arrow[r, dashed, "\sigma_1"] & \Delta_1 \end{tikzcd} \qquad \begin{tikzcd} & \Gamma \arrow[dr, "\sigma_1"] & & \\ A_2 \arrow[ur, "t_2"] \arrow[dr, "u_2"'] & & \Delta_1 \arrow[r, dashed, "\sigma_2"] & \Delta_2 \\ & \Gamma \arrow[ur, "\sigma_1"'] & & \end{tikzcd}$$

---

$$\begin{tikzcd} A_1 + A_2 \arrow[r, shift left, "{t_1, t_2}"] \arrow[r, shift right, "{u_1, u_2}"'] & \Gamma \arrow[r, dashed, "\sigma_2 \circ \sigma_1"] & \Delta_2 \end{tikzcd}$$

# Soundness of U-FlexFlex

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

Diagrammatically,

$$\dfrac{a \overset{x}{\underset{y}{\rightrightarrows}} b \overset{z}{\dashrightarrow} c \qquad \text{in } \mathcal{A}^{op}}{\mathcal{L}a \overset{\mathcal{L}x}{\underset{\mathcal{L}y}{\rightrightarrows}} \mathcal{L}b \overset{\mathcal{L}z}{\dashrightarrow} \mathcal{L}c \quad \text{in } \mathrm{Th}^{op}(F)}$$

where

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

# Soundness of U-NoCycle

$$\boxed{\dfrac{u_{|\Gamma} = \underline{u'} \qquad \Gamma \vdash u' :> M(x) \Rightarrow v; \sigma \dashv \Delta}{\Gamma, M : b \vdash M(x) = u \Rightarrow \sigma, M \mapsto v \dashv \Delta}\textsc{U-NoCycle}}$$

Diagrammatically,

$$\begin{tikzcd} ya \arrow[r, "{"M(x)"}"] \arrow[d, "u'"'] & yb \arrow[d, dashed, "v"] \\ \Gamma \arrow[r, dashed, "\sigma"'] & \Delta \end{tikzcd}$$

pushout

---

$$\begin{tikzcd} & yb \arrow[dr, "in_2"] & & \\ ya \arrow[ur, "{"M(x)"}"] \arrow[dr, "u'"'] & & \Gamma + yb \arrow[r, dashed, "{[v,\sigma]}"] & \Delta \\ & \Gamma \arrow[ur, "in_1"'] & & \end{tikzcd}$$

coequaliser

# Outline

# Types

## Notation

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

$$\frac{1 \leq i \leq n}{n \vdash \eta_i \text{ type}}\textsc{Type-Var} \qquad \frac{n+1 \vdash \tau \text{ type}}{n \vdash \forall\tau \text{ type}}\textsc{Forall}$$

$$\frac{n \vdash \tau_1, \tau_2 \text{ type}}{n \vdash \tau_1 \to \tau_2 \text{ type}}\textsc{Arrow}$$

# Metavariable arities

## Metavariable application

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

$$M : (p | \overbrace{\underbrace{\tau_1, \ldots, \tau_q}_{\text{input argument types}} \vdash \tau_f}^{p \vdash \tau_i \text{ type}})$$

number of type variable arguments

output type

# Signature

- Objects of $\mathcal{A}$ = metavariables arities

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

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

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

# The endofunctor for System $F$

| Typing rule | $F(X)_{n\|\vec{\tau}\vdash\sigma_f} = \coprod \ldots$ |
|---|---|
| $\dfrac{\tau_i = \sigma_f}{n\|\vec{\tau} \vdash v_i : \sigma_f}\textsc{Var}$ | $\|\vec{\tau}\|_{\sigma_f}$ |
| $\dfrac{n\|\vec{\tau}, \sigma_1 \vdash t : \sigma_2}{n\|\vec{\tau} \vdash \lambda t : \sigma_1 \to \sigma_2}\textsc{Abs}$ | $\coprod_{\sigma_1,\sigma_2 \text{ s.t. } \sigma_f = (\sigma_1 \to \sigma_2)} X_{n\|\vec{\tau},\sigma_1\vdash\sigma_2}$ |
| $\dfrac{n\|\vec{\tau} \vdash t : \sigma \to \sigma_f \quad n\|\vec{\tau} \vdash u : \sigma_f}{n\|\vec{\tau} \vdash t\ u : \sigma_f}\textsc{App}$ | $\coprod_{\sigma} X_{n\|\vec{\tau}\vdash\sigma\to\sigma_f} \times X_{n\|\vec{\tau}\vdash\sigma}$ |
| $\dfrac{n+1\|\vec{\tau} \vdash t : \sigma}{n\|\vec{\tau} \vdash \Lambda t : \forall\sigma}\textsc{T-Abs}$ | $\coprod_{\sigma \text{ s.t. } \sigma_f=\forall\sigma} X_{n+1\|\vec{\tau}\vdash\sigma}$ |
| $\dfrac{n\|\vec{\tau} \vdash t : \forall\sigma_1}{n\|\vec{\tau} \vdash t \cdot \sigma_2 : \sigma_1[\sigma_2]}\textsc{T-App}$ | $\coprod_{\sigma_1,\sigma_2 \text{ s.t. } \sigma_f=\sigma_1[\sigma_2]} X_{n\|\vec{\tau}\vdash\forall\sigma_1}$ |

# Unification in system F: an example

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

## Most general unifier

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

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

$$\frac{}{\Gamma \vdash () = () \Rightarrow id_\Gamma \dashv \Gamma} \qquad \frac{}{\bot \vdash \vec{t} = \vec{u} \Rightarrow ! \dashv \bot}$$

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

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

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

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

$$\frac{u = o(\vec{t}) \qquad u_{|\Gamma} \neq \underline{\ldots}}{\Gamma, M : b \vdash M(x) = u \Rightarrow ! \dashv \bot}\text{U-CYCLIC} \quad + \text{sym}$$

## Non cyclic phase

$$\frac{}{\Gamma \vdash () :> () \Rightarrow (); id_\Gamma \dashv \Gamma} \qquad \frac{}{\bot \vdash \vec{t} :> \vec{f} \Rightarrow !; ! \dashv \bot}$$

$$\frac{\Gamma \vdash t_1 :> M_1(\vec{x}) \Rightarrow u_1; \sigma_1 \dashv \Delta_1 \quad \Delta_1 \vdash \vec{t_2}[\sigma_1] :> \vec{M_2} \Rightarrow \vec{u_2}; \sigma_2 \dashv \Delta_2}{\Gamma \vdash t_1, \vec{t_2} :> M_1(\vec{x}), \vec{M_2} \Rightarrow u_1[\sigma_2], \vec{u_2}; \sigma_1[\sigma_2] \dashv \Delta_2}$$

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

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

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

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

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

$$\frac{\Gamma \vdash \overbrace{\vec{t}}^{()} :> \ldots \qquad \overbrace{o = \vec{x} \cdot o'}^{o = x_{o'}}}{\Gamma \vdash o(\vec{t}) :> N(\vec{x}) \Rightarrow o'(\vec{u}); \sigma \dashv \Delta} \quad \Leftrightarrow \quad \frac{o = x_i}{\Gamma \vdash o :> N(\vec{x}) \Rightarrow i; id_\Gamma \dashv \Gamma}$$

$$\frac{o \neq x \cdot \ldots}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow !; ! \dashv \bot} \quad \Leftrightarrow \quad \frac{o \notin \vec{x}}{\Gamma \vdash o :> N(\vec{x}) \Rightarrow !; ! \dashv \bot}$$

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

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

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

$$\frac{o \neq x \cdot \ldots}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow !; ! \dashv \bot} \qquad \text{never applies}$$

# Typing rule for metavariables

## Typing judgement

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

## Typing metavariables

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

# Future directions

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