# Generic pattern unification A categorical approach

Ambroise Lafont Neel Krishnaswami

University of Cambridge

October 2022

### What is unification?

$$t \stackrel{?}{=} u$$
  
terms with metavariables  $M, N, \dots$ 

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

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

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

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

**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

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

### Second-order unification

 $M(\dots)$ 

### Example

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:

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

∃ unification algorithm [Miller '91]

- fails if no unifier
- returns the most general unifier

### 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

## 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

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

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

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

## Syntax (De Bruijn levels)

Metavariable context 
$$(M_1:n_1,\dots)$$

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

$$\frac{x < n}{\Gamma; \, n \vdash x} \mathrm{VAR} \qquad \frac{\Gamma; \, n \vdash t \quad \Gamma; \, n \vdash u}{\Gamma; \, n \vdash t \quad u} \mathrm{APP} \qquad \frac{\Gamma; \, n + 1 \vdash t}{\Gamma; \, n \vdash \lambda t} \mathrm{ABS}$$

$$\frac{(M:n) \in \Gamma \qquad x_1, \dots, x_n < n \qquad x_1, \dots x_n \text{ distinct}}{\Gamma; n \vdash M(x_1, \dots, x_n)} \text{FLEX}$$

### Metavariable substitution

Substitution 
$$\sigma$$
 from  $(M_1: m_1, \ldots, M_p: m_p)$  to  $\Delta$ :

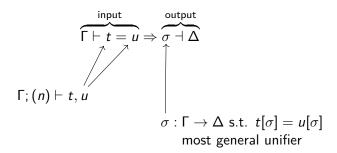
$$(\sigma_1,\ldots,\sigma_p)$$
 s.t.  $\Delta; m_i \vdash \sigma_i$ 

 $\sigma$  extends to terms:

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

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

## Unification algorithm



## **Examples**

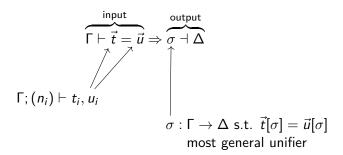
$$\Gamma, M: 2 \vdash M(5,3) = 5 \Rightarrow (M \mapsto 0) \dashv \Gamma$$

$$\Gamma, M: 2 \vdash M(5,3) = 3 \Rightarrow (M \mapsto 1) \dashv \Gamma$$

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

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

## Unifying lists of terms

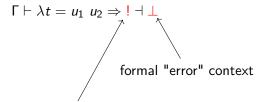


## Examples (lists)

$$\Gamma \vdash () = () \Rightarrow id_{\Gamma} \dashv \Gamma$$

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

## Impossible cases



formal "error" substitution

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

### Three cases

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

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

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

## Unifying a metavariable with itself

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

Most general unifier:  $M \mapsto M'(\vec{z})$ 

•  $\vec{z} = \text{vector of common positions: } x_{\vec{z}} = y_{\vec{z}}$ 

Formally,

$$\frac{"n \vdash \vec{x} = \vec{y} \Rightarrow \vec{z} \dashv p"}{\Gamma, M : n \vdash M(\vec{x}) = M(\vec{y}) \Rightarrow M \mapsto M'(\vec{z}) \dashv \Gamma, M' : p}$$

## Cyclic case

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

No unifier

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

sizes cannot match after substitution

## Non cyclic case

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

Most general unifier:  $M \mapsto u[\vec{x}^{-1}]$ 

Requires

$$fv(u) \subset \vec{x}$$
 (1)

 $\Rightarrow$  **Pruning phase**: enforces (1) by restricting metavariable arities.

### Example

$$M(x)$$
  $\stackrel{?}{=}$   $N(x,y)$   
 $N(x,y)$   $\xrightarrow{\text{pruning}}$   $N'(x)$ 

## Pruning phase

$$u$$
 after pruning and renamed by  $\vec{x}^{-1}$   $\Gamma \vdash u :> M(\vec{x}) \Rightarrow v$ ;  $\sigma \dashv \Delta$   $M \notin \Gamma$ ,  $u$   $\sigma : \Gamma \rightarrow \Delta$  pruning substitution

### Intuition

$$u \stackrel{?}{=} M(\vec{x}) \quad \Rightarrow \quad (\sigma, M \mapsto v) = \text{most general unifier}$$

## Pruning a metavariable

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

**Most general unifier**:  $M \mapsto N'(\vec{l})$ ,  $N \mapsto N'(\vec{r})$  such that

$$x_{\vec{l}} = y_{\vec{r}}$$

$$\frac{"n \vdash \vec{x} :> \vec{y} \Rightarrow \vec{l}; \vec{r} \dashv p"}{\Gamma, N : n \vdash N(\vec{x}) :> M(\vec{y}) \Rightarrow N'(\vec{l}); N \mapsto N'(\vec{r}) \dashv \Gamma, N' : p}$$

## Pruning: other examples

$$\frac{y \notin \vec{x}}{\Gamma \vdash x_{i} :> M(x_{0}, \dots, x_{n}) \Rightarrow i; id_{\Gamma} \dashv \Gamma} \qquad \frac{y \notin \vec{x}}{\Gamma \vdash y :> M(\vec{x}) \Rightarrow !; ! \dashv \bot}$$
bound variable
$$\frac{\Gamma \vdash t :> M_{1}(\vec{x}, \cap) \Rightarrow v; \sigma \dashv \Delta}{\Gamma \vdash \lambda t :> M(\vec{x}) \Rightarrow \lambda v; \sigma \dashv \Delta}$$

$$\frac{\text{"}\Gamma \vdash t, u :> M_1(\vec{x}), M_2(\vec{x}) \Rightarrow v_1, v_2; \sigma \dashv \Delta\text{"}}{\Gamma \vdash t \ u :> M(\vec{x}) \Rightarrow v_1 \ v_2; \sigma \dashv \Delta}$$

## Pruning multi-terms

$$\Gamma \vdash u_1, \ldots, u_n :> M_1(\vec{x}_1), \ldots M_n(\vec{x}_n) \Rightarrow v_1, \ldots, v_n; \sigma \dashv \Delta$$

$$\Gamma; (n_i) \vdash u_i \qquad \Delta; m_i \vdash v_i$$

$$(M_i : m_i) \notin \Gamma, u_i \qquad \sigma : \Gamma \rightarrow \Delta$$

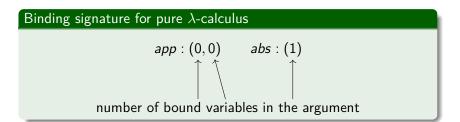
### Rules for multi-terms

$$\Gamma \vdash () :> () \Rightarrow (); id_{\Gamma} \dashv \Gamma$$

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

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

## Parameterisation by a signature



## Example: pruning an operation

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

$$\frac{\Gamma \vdash \vec{t} :> M_1(\vec{x}, \overbrace{n, \dots, n + \alpha_1 - 1}), \dots, M_p(\dots) \Rightarrow \vec{u}; \sigma \dashv \Delta}{\Gamma \vdash o(\vec{t}) :> N(\vec{x}) \Rightarrow o(\vec{u}); \sigma \dashv \Delta}$$

- 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

- 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

### Pure $\lambda$ -calculus as a functor

category of finite cardinals and injections between them



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

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

## Pure $\lambda$ -calculus as a fixpoint

$$\Lambda_n \cong \underbrace{\{0,\ldots,n-1\}}_{\text{variables}} + \underbrace{\Lambda_n \times \Lambda_n}_{\text{application}} + \underbrace{\Lambda_{n+1}}_{\text{abstraction}}$$

In fact,

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

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

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

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

$$\Lambda(m)_n = \{t \mid M : m; n \vdash t\}$$

As a fixpoint:

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

$$arg^{M}_{n} = \{M\text{-arguments in the variable context } n\}$$

$$= \{\vec{x} \in \{0, \dots, n-1\}^{m} \mid x_{1}, \dots, x_{m} \text{ distinct } \}$$

$$= \text{hom}_{\mathbb{F}_{n}}(m, n)$$

$$\Lambda(m) = \mu X.(F(X) + ym)$$

### Pure $\lambda$ -calculus with metavariables

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

As a fixpoint:

$$\Lambda(\Gamma) = \mu X.(F(X) + \underbrace{\coprod_{(M:m) \in \Gamma} ym}_{\underline{\Gamma}})$$

$$= \underbrace{T}_{\text{free monad generated by } F} (\underline{\Gamma})$$

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

## Unification as a Kleisli coequaliser

#### Claims:

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

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

### 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

### Signature

- lacktriangledown  $\mathcal A$  small category (e.g.,  $\mathbb F_m$ , 1)
- **Intuition:** objects = metavariable arities, morphisms = metavariable arguments.
  - All morphisms in A are monic (pattern restriction).
  - $\mathcal{A}$  has finite connected limits  $(M(\vec{x}) \stackrel{?}{=} N(\vec{y}))$ .
  - $oldsymbol{2}$  F endofunctor on  $[\mathcal{A}, \operatorname{Set}]$  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

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

$$F(X)_{a} = \coprod_{o \in O_{a}} X_{L_{o,1}} \times \cdots \times X_{L_{o,n_{o}}}$$

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

### Semantics of unification

**Claim**: Given a signature (A, F), a coequaliser diagram in Th(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**

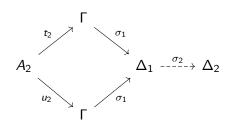
mostly used in  $Th(F)_{\perp} = Th(F) + a$  free terminal object  $\perp$ .

# 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,

$$A_1 \xrightarrow[u_1]{t_1} \Gamma \xrightarrow{--\sigma_1} \Delta_1$$



$$A_1 + A_2 \xrightarrow[H_1, H_2]{t_1, t_2} \Gamma \xrightarrow{\sigma_2 \circ \sigma_1} \Delta_2$$

### Soundness of U-FLEXFLEX

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

Diagrammatically,

$$a \xrightarrow{x} b - \xrightarrow{z} c \quad \text{in } \mathcal{A}^{op}$$

$$\mathcal{L}a \xrightarrow{\mathcal{L}x} \mathcal{L}b - \xrightarrow{\mathcal{L}z} \mathcal{L}c \quad \text{in } \mathsf{Th}(F)$$

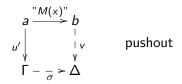
where

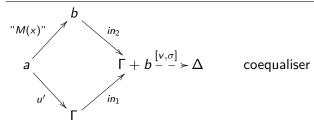
$$a \xrightarrow{x} b \xrightarrow{\mathcal{L} : \mathcal{A}^{op} \to \mathsf{Th}(F)} ya \xrightarrow{"M(x)"} T(\underline{M : b})$$

### Soundness of U-NoCycle

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

#### Diagrammatically,





### Outline

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

# **Types**

#### Notation

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

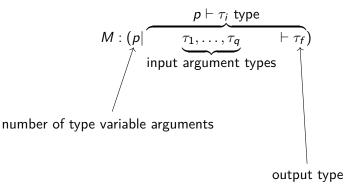
$$\frac{\alpha < n}{n \vdash \alpha \text{ type}} \text{Type-Var} \qquad \frac{n+1 \vdash \tau \text{ type}}{n \vdash \forall \tau \text{ type}} \text{Forall}$$

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

#### Metavariable arities

#### Metavariable application

type variables "ground" variables 
$$M(\alpha_1, \dots, \alpha_p \mid x_1, \dots, x_q)$$



## Typing rule for metavariables

#### Typing judgement

Type variable context | Types of variables | 
$$\underbrace{T}_{n}$$
 |  $\underbrace{t_1, \dots, t_m}_{n \vdash t_i \text{ type}} \vdash u : t_f$ 

#### Typing metavariables

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

## Signature

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

$$n|\underbrace{\tau_1,\ldots,\tau_p}_C \vdash \tau_f$$

• Need an endofunctor F on [A, Set] s.t.

$$\mu F(n|C \vdash \tau_f) = \{t \text{ s.t. } n|C \vdash t : \tau_f\}$$

# The endofunctor for System F

Typing rule	$F(X)_{n\mid C\vdash \tau}=\coprod\ldots$
$\frac{C_i = \tau}{n C \vdash i : \tau} \text{VAR}$	$ C _{ au}$
$\frac{n C, \tau_1 \vdash t : \tau_2}{n C \vdash \lambda t : \tau_1 \to \tau_2} ABS$	$\coprod_{\tau_1,\tau_2 \text{ s.t. } \tau = (\tau_1 \to \tau_2)} X_{n C,\tau_1 \vdash \tau_2}$
$\frac{n C \vdash t : \tau' \to \tau  n C \vdash u : \tau'}{n C \vdash t \ u : \tau} APP$	$\coprod_{\tau'} X_{n C\vdash \tau'\to \tau} \times X_{n C\vdash \tau'}$
$\frac{n+1 C \vdash t : \tau'}{n C \vdash \Lambda t : \forall \tau'} \text{T-Abs}$	$\coprod_{\tau' \text{ s.t. } \tau = \forall \tau'} X_{n+1 C \vdash \tau'}$
$\frac{n C \vdash t : \forall \tau_1}{n C \vdash t \cdot \tau_2 : \tau_1[\tau_2]} \text{T-App}$	$\coprod_{\tau_1,\tau_2 \text{ s.t. } \tau=\tau_1[\tau_2]} X_{n C\vdash \forall \tau_1}$

### Unification in system F: an example

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

**Most general unifier**:  $M \mapsto N(\vec{\gamma}, \vec{z})$ , where

•  $\vec{\gamma}$  maximal s.t.

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

•  $\vec{z}$  maximal s.t.

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

# Summary of the generic unification algorithm

$$\frac{\Gamma \vdash () = () \Rightarrow id_{\Gamma} \dashv \Gamma}{\Gamma \vdash t_{1} = u_{1} \Rightarrow \sigma_{1} \dashv \Delta_{1}} \qquad \frac{\bot \vdash \vec{t} = \vec{u} \Rightarrow ! \dashv \bot}{\Gamma \vdash t_{1} = u_{1} \Rightarrow \sigma_{1} \dashv \Delta_{1}} \qquad \Delta_{1} \vdash \vec{t_{2}}[\sigma_{1}] = \vec{u_{2}}[\sigma_{1}] \Rightarrow \sigma_{2} \dashv \Delta_{2}} \text{U-SPLIT}$$

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

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

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

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

# Pruning phase

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

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

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

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

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

Examples in next slides

◆□ ▶ ◆● ▶ ◆ ■ ▶ ◆ ■ り へ ○ 53/55

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

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

$$\frac{\Gamma \vdash \overrightarrow{t} :> \dots \underbrace{o = \overrightarrow{x} \cdot o'}_{o = \overrightarrow{x} \cdot o'}}{\Gamma \vdash o(\overrightarrow{t}) :> N(\overrightarrow{x}) \Rightarrow o'(\overrightarrow{u}); \sigma \dashv \Delta} \Leftrightarrow \frac{o = x_{i}}{\Gamma \vdash o :> N(\overrightarrow{x}) \Rightarrow i; id_{\Gamma} \dashv \Gamma}$$

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

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

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

$$\frac{\Gamma \vdash \vec{t} :> \overbrace{\mathcal{L}^{+} x^{o}}^{M_{1}(\vec{x}, n, \dots, n + \alpha_{1} - 1), \dots, M_{p}(\dots)} \Rightarrow \vec{u}; \sigma \dashv \Delta}{\Gamma \vdash o(\vec{t}) :> N(x) \Rightarrow o'(\vec{u}); \sigma \dashv \Delta}$$

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