Generic pattern unification

Ambroise Lafont Neel Krishnaswami

University of Cambridge

A quick introduction to unification

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

terms with metavariables $M, N, \ldots$

Unifier = metavariable substitution σ s.t.

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

Most general unifier = unifier σ 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

A quick introduction to unification

$$\underbrace{t}\overset{?}{=}\underbrace{u}$$
terms with metavariables $M, N, \ldots$

Unifier $=$ metavariable substitution σ s.t.

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

Most general unifier $=$ unifier σ 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

- 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:

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

$\exists$ unification algorithm [Miller '91]

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

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 custom notion of *signature*
- Categorical semantics

Examples

- *binding signatures*
- Ordered syntax
- 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. Generalised binding signatures
3. Categorical semantics
   - 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. Generalised binding signatures
3. Categorical semantics
   - 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. Generalised binding signatures
3. Categorical semantics
   - A case study: syntax of pure λ-calculus
   - Generic pattern unification
4. Example: System F

Syntax (De Bruijn levels)

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

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

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

No β/η-equation.

Metavariable substitution

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

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

Notation

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

Term substitution

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

Base case:

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

Metavariable substitution

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

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

Notation

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

Term substitution

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

Base case:

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

Outline

1. Pattern unification for pure λ-calculus
   - Syntax
   - Unification algorithm
2. Generalised binding signatures
3. Categorical semantics
   - A case study: syntax of pure λ-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

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

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

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

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

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

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

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

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

sizes cannot match after substitution

Non-cyclic case

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

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 $(x \neq y)$

Non-cyclic case

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

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 ($x \neq y$)

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

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.

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

Pruning

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

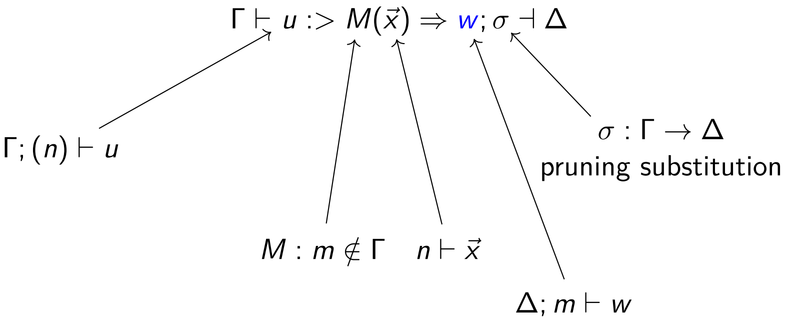
Most general unifier

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

- 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



Formal meaning

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

Pruning a variable

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

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

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:

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

Then,

$$M(v_1, \ldots, v_m) \mapsto P(v_{l_1}, \ldots, v_{l_p})$$
$$N(v_1, \ldots, v_n) \mapsto P(v_{r_1}, \ldots, v_{r_p})$$

Examples

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

Pruning a metavariable

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

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

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

Pruning operations

$$o(\vec{t}) \stackrel{?}{=} M(\vec{x}) \qquad (M \notin \vec{t})$$

Divide & Conquer: a fresh metavariable for each argument.

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

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

Pruning operations

$$o(\vec{t}) \stackrel{?}{=} M(\vec{x}) \qquad (M \notin \vec{t})$$

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

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

$\Gamma; (n_i) \vdash u_i$

$\Delta; m_i \vdash w_i$

$(M_i : m_i) \notin \Gamma$

$\sigma : \Gamma \to \Delta$

Formal meaning

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

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

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 λ-calculus

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

number of bound variables in the argument

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

$$o : (\overline{o}_1, \dots, \overline{o}_p)$$

$$\frac{\Gamma \vdash \vec{t} :> M_1(\vec{x}, \overbrace{v_{n+1}, \dots, v_{n+1+\overline{o}_1}}^{\text{bound variables}}), \dots, M_p(\dots) \Rightarrow \vec{u}; \sigma \dashv \Delta}{\Gamma \vdash o(\vec{t}) :> M(\vec{x}) \Rightarrow o(\vec{u}); \sigma \dashv \Delta}$$

Generalised binding signatures

Generalised binding signatures = our notion of signature for syntax with (pattern-restricted) metavariables.

Examples

- (Simply-typed) binding signatures
- Linearly ordered λ-calculus
- Intrinsic System F

Pruning operations

$$\frac{\Gamma \vdash \vec{t} :> M_1(x_1^{o'}), \ldots, M_n(x_n^{o'}) \Rightarrow \vec{u}; \sigma \dashv \Delta \qquad o = o'\{x\}}{\Gamma \vdash o(\vec{t}) :> M(x) \Rightarrow o'(\vec{u}); \sigma \dashv \Delta}\text{P-RIG}$$

$$\frac{o \neq \ldots\{x\}}{\Gamma \vdash o(\vec{t}) :> M(x) \Rightarrow !; ! \dashv \bot}\text{P-FAIL}$$

Outline

Outline

Pure λ-calculus as a functor

category of finite cardinals and injections between them

↓

Pure λ-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

$$\Lambda_n \cong \underbrace{\{v_1, \ldots, v_n\}}_{\text{variables}} + \underbrace{\Lambda_n \times \Lambda_n}_{\text{application}} + \underbrace{\Lambda_{n+1}}_{\text{abstraction}}$$

In fact,

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

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

$$F(X)_n = \{v_1, \ldots, v_n\} + X_n \times X_n + X_{n+1}$$

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

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

As an initial algebra:

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

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

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

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

As an initial algebra:

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

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

Pure λ-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 λ-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 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

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 $\mathrm{MCon}(F) \subset Kl(T)$.

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

$\mathrm{MCon}(F)^{op}$ is a "non-free" Lawvere theory

[1] well-known in the first-order case ('free' Lawvere theories).

Outline

Endofunctor generated by a signature

A GB-signature $(\mathcal{A}, O, \alpha)$ generates an endofunctor on $[\mathcal{A}, \mathrm{Set}]$ of the shape

$$F(X)_a = \coprod_{o \in O_n(a)} X_{\overline{o}_1} \times \cdots \times X_{\overline{o}_n}$$

Let T denote the free monad generated by F.

$$\underbrace{\Gamma}_{M_1 : m_1, \ldots, M_n : m_n} ; a \vdash t \qquad \text{means} \qquad t \in T(\underbrace{\Gamma}_{ym_1 + \cdots + ym_n})_a$$

Endofunctor generated by a signature

A GB-signature $(\mathcal{A}, O, \alpha)$ generates an endofunctor on $[\mathcal{A}, \mathrm{Set}]$ of the shape

$$F(X)_a = \coprod_{o \in O_n(a)} X_{\overline{o}_1} \times \cdots \times X_{\overline{o}_n}$$

Let T denote the free monad generated by F.

$$\underbrace{\Gamma}_{M_1 : m_1, \ldots, M_n : m_n}; a \vdash t \qquad \text{means} \qquad t \in T(\underbrace{\underline{\Gamma}}_{ym_1 + \cdots + ym_n})_a$$

Generic unification algorithm

Conditions

1. All morphisms in $\mathcal{A}$ are mono (pattern restriction)
2. $\mathcal{A}$ has equalisers and pullbacks (Cf $\mathbb{F}_m$)
3. If $X : \mathcal{A} \to \mathrm{Set}$ preserves them, then $F(X)$ also does.

Claim: A coequaliser diagram in $\mathrm{MCon}(F)$ has a colimit as soon as there exists a cocone (i.e., a 'unifier').

Getting rid of partiality

Equivalent claim:

$$\underbrace{\mathrm{MCon}_{\perp}(F)}_{\mathrm{MCon}(F)\text{ extended with a free terminal object }\perp} \text{ has coequalisers.}$$

Proof: By describing a unification algorithm, constructing coequalisers in $\mathrm{MCon}_{\perp}(F)$.

Generic unification algorithm

Conditions

1. All morphisms in $\mathcal{A}$ are mono (pattern restriction)
2. $\mathcal{A}$ has equalisers and pullbacks (Cf $\mathbb{F}_m$)
3. If $X : \mathcal{A} \to \mathrm{Set}$ preserves them, then $F(X)$ also does.

Claim: A coequaliser diagram in $\mathrm{MCon}(F)$ has a colimit as soon as there exists a cocone (i.e., a 'unifier').

Getting rid of partiality

Equivalent claim:

$$\underbrace{\mathrm{MCon}_{\perp}(F)}_{\mathrm{MCon}(F)\text{ extended with a free terminal object }\perp} \text{ has coequalisers.}$$

Proof: By describing a unification algorithm, constructing coequalisers in $\mathrm{MCon}_{\perp}(F)$.

Generic unification algorithm

Conditions

1. All morphisms in $\mathcal{A}$ are mono (pattern restriction)
2. $\mathcal{A}$ has equalisers and pullbacks (Cf $\mathbb{F}_m$)
3. If $X : \mathcal{A} \to \mathrm{Set}$ preserves them, then $F(X)$ also does.

Claim: A coequaliser diagram in $\mathrm{MCon}(F)$ has a colimit as soon as there exists a cocone (i.e., a 'unifier').

Getting rid of partiality

Equivalent claim:

$$\underbrace{\mathrm{MCon}_\perp(F)}_{\mathrm{MCon}(F) \text{ extended with a free terminal object } \perp} \text{ has coequalisers.}$$

Proof: By describing a unification algorithm, constructing coequalisers in $\mathrm{MCon}_\perp(F)$.

Interpreting the unification statements

Notations

$$\Gamma \vdash t = u \Rightarrow \sigma \dashv \Delta \quad \Leftrightarrow \quad \cdot \underset{u}{\overset{t}{\rightrightarrows}} \Gamma \overset{\sigma}{\dashrightarrow} \Delta \qquad \text{coequaliser}$$

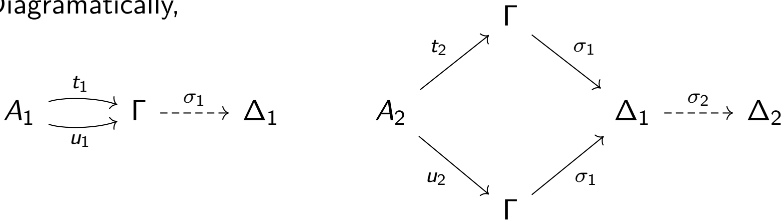
$$\Gamma \vdash t :> f \Rightarrow v ; \sigma \dashv \Delta \quad \Leftrightarrow \quad \begin{array}{ccc} \cdot & \xrightarrow{f} & \cdot \\ {\scriptstyle t}\downarrow & & \downarrow{\scriptstyle v} \\ \Gamma & \underset{\sigma}{\dashrightarrow} & \Delta \end{array} \qquad \text{pushout}$$

mostly used in $\mathrm{MCon}(F)_\perp$.

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

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

Diagramatically,



$$A_1 + A_2 \underset{u_1, u_2}{\overset{t_1, t_2}{\rightrightarrows}} \Gamma \overset{\sigma_2 \circ \sigma_1}{\dashrightarrow} \Delta_2$$

Soundness of U-FlexFlex

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

Diagrammatically,

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

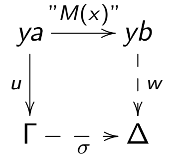
where

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

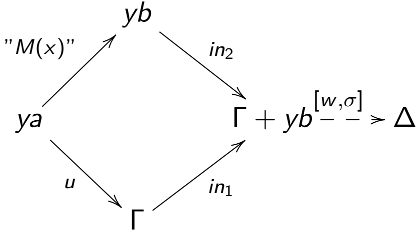
Soundness of U-NoCycle

$$\frac{M \notin u \qquad \Gamma \vdash u :> M(x) \Rightarrow w; \sigma \dashv \Delta}{\Gamma, M : m \vdash M(x) = u \Rightarrow \sigma, M \mapsto w \dashv \Delta}\textsc{U-NoCycle}$$

Diagrammatically,



pushout



coequaliser

Outline

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

Types

Notation

$n \vdash \tau$ type $\quad\Leftrightarrow\quad$ the type τ 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

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

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

p: number of type variable arguments

τ_f: output type

Signature for System F

Objects of $\mathcal{A}$ are of the shape $n|\vec{\tau} \vdash \sigma_f$

Typing rule	$O_p(n\|\vec{\sigma} \vdash \tau) = \coprod \ldots$	$\alpha_o = (\ldots)$
$\dfrac{x : \tau \in \Gamma}{n\|\Gamma \vdash x : \tau}$	$\{v_i \text{ s.t. } i \in \|\vec{\sigma}\|_\tau\}$	$()$
$\dfrac{n\|\Gamma \vdash t : \tau' \Rightarrow \tau \quad n\|\Gamma \vdash u : \tau'}{n\|\Gamma \vdash t\ u : \tau}$	$\{a_{\tau'} \text{ s.t. } n \vdash \tau' \text{ type}\}$	$\begin{pmatrix} n\|\vec{\sigma} \to \tau' \Rightarrow \tau \\ n\|\vec{\sigma} \to \tau' \end{pmatrix}$
$\dfrac{n\|\Gamma, x : \tau_1 \vdash t : \tau_2}{n\|\Gamma \vdash \lambda x.t : \tau_1 \Rightarrow \tau_2}$	$\{l_{\tau_1,\tau_2} \text{ s.t. } \tau = (\tau_1 \Rightarrow \tau_2)\}$	$(n\|\vec{\sigma}, \tau_1 \to \tau_2)$
$\dfrac{n\|\Gamma \vdash t : \forall\tau_1 \quad \tau_2 \in S_n}{n\|\Gamma \vdash t \cdot \tau_2 : \tau_1[\tau_2]}$	$\{A_{\tau_1,\tau_2} \text{ s.t. } \tau = \tau_1[\tau_2]\}$	$(n\|\vec{\sigma} \to \forall\tau_1)$
$\dfrac{n+1\|wk(\Gamma) \vdash t : \tau}{n\|\Gamma \vdash \Lambda t : \forall\tau}$	$\{\Lambda_{\tau'} \text{ s.t. } \tau = \forall\tau'\}$	$(n+1\|wk(\vec{\sigma}) \to \tau')$

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

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

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

Future directions

- Mecanisation (needs rephrasing using structural recursion)
- Dependent types
- Unification modulo reduction

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

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