Higher-order Unification from E-unification with 2nd-order equalities and parametrised metavariables

Nikolai Kudasov
Innopolis University

Implementing a Dependently Typed Language

(Motivation)

Components

① Syntax			
② Evaluation	requires	Substitution	(easy)
③ Type checker	may rely on	First-Order Unification	(easy)
④ Type inference	can be reduced to (Mazzoli & Abel, 2016)	Higher-Order Unification	(hard)

In practice, it is often good to work with syntax schemas, and generate actual syntax together with substitution and FOU.

Syntax schema (e.g. generating functor)

Syntax + substitution + FOU

(e.g. as a free monad) (Swierstra, 2008)

(via datatype-generic programming)

Problem: first-order syntax does not carry enough information!

1. bound variables have to be processed (e.g. de Brujn indices)
2. scopes are built into evaluation, not inferred from syntax
3. higher-order unification has to rely on particular presentation of functions and applications (not generic)

Idea: use 2nd-order syntax! (see also Fiore & Szamozvancev, 2022)

1. Bound variables and scopes are explicit
2. Can be implemented using nested datatypes (Bird & Patterson, 1999)
3. Given evaluator, higher-order unification comes for free*! (Kudasov, arxiv:2204.05653)

* under certain assumptions about evaluator

2nd-order syntax with parametrised metavariables

(Fiore, 2008)

Definition 1. A term is

1. a variable $x, y, z, \dots$
2. an applied function symbol $F(t_1, \dots, t_p, \underbrace{x.s_1, \dots, x.s_q}_{\text{scoped subterms}})$
3. a parametrised metavariable $m[t_1, \dots, t_k]$

Example 1. Here are some examples of terms:

$\mathsf{APP}(\mathsf{LAM}(x.\,m_1[x]),\ m_2[\,])$ $\qquad (\lambda x.\, m_1[x])\, m_2[\,]$

$m[\mathsf{PAIR}(\mathsf{LAM}(f.\ \mathsf{APP}(f, y)),\ g)]$ $\qquad m[\langle \lambda f.\, f y, g\rangle]$

$m_1[m_2[\,]]$

Definition 2. We define variable substitution as usual, and meta substitution as follows:

① $$x\,[m[\overline{z_k}] \mapsto u] \;:=\; x$$

② $$F(\overline{t_p},\ \overline{x.s_q})[m[\overline{z_k}] \mapsto u] \;:=\; F\big(\overline{t_p[m[\overline{z_k}]\mapsto u]},\ \overline{x.s_q[m[\overline{z_k}]\mapsto u]}\big)$$

③ $$m[\overline{t_k}]\,[m[\overline{z_k}]\mapsto u] \;:=\; u[\overline{z_k \mapsto t_k}]$$

④ $$m'[\overline{t_n}]\,[m[\overline{z_k}]\mapsto u] \;:=\; m'[\,\overline{t_n[m[\overline{z_k}]\mapsto u]}\,] \qquad m \neq m'$$

E^2-unification

(Fiore & Mahmoud, 2013)

Definition 3. A 2nd-order equality is a pair of terms in a context:

$$\underbrace{x_1, x_2, \ldots, x_n}_{\text{free variables}},\ \underbrace{m_1 : k_1, \ldots, m_\ell : k_\ell}_{\text{metavariables w/ arities}} \ \vdash\ t_1 \equiv t_2$$

Example 2. We can encode β-equalities of λ-calculus with pairs:

① $$(\lambda x.\, t)\, u \equiv_\beta t[x \mapsto u]$$

$$m_1 : 1,\ m_2 : 0 \ \vdash\ \mathsf{APP}(\mathsf{LAM}(x.\, m_1[x]),\ m_2[\,]) \equiv m_1[m_2[\,]]$$

② $\pi_1 \langle t, u \rangle \equiv_\beta t$

$m_1 : 0,\ m_2 : 0 \vdash \mathsf{FIRST}(\mathsf{PAIR}(m_1[], m_2[])) \equiv m_1[]$

③ $\pi_2 \langle t, u \rangle \equiv_\beta u$

$m_1 : 0,\ m_2 : 0 \vdash \mathsf{SECOND}(\mathsf{PAIR}(m_1[], m_2[])) \equiv m_2[]$

Definition 4. A 2nd-order constraint is a pair of terms in a context:

$$\underbrace{\forall \overline{x_n}.}_{\text{bound variables}}\ t_1 \stackrel{?}{=} t_2$$

Definition 5. Given

- a set E of 2nd-order equalities,
- a set C of 2nd-order constraints

we call $\langle E, C \rangle$ an E^2-unification problem.

A substitution $\sigma = [m_1[\overline{z_{k_1}}] \mapsto t_1, \ldots, m_\ell[\overline{z_{k_\ell}}] \mapsto t_\ell]$ is called a solution to the E^2-unification problem $\langle E, C \rangle$, if for all constraints $\forall \overline{x_n}.\ u_1 \stackrel{?}{=} u_2$ in C we have

$$\overline{x_n} \vdash \sigma u_1 \equiv \sigma u_2 \quad (\text{modulo } E)$$

Claim 1. Let $f_1, \ldots, f_n$ be (non-parametrised) metavariables. Let $C_{HOU} = \{ t_1 \stackrel{?}{=} t_2, \ldots \}$ be a higher-order unification problem over F_n for the untyped λ-calculus. Then C_{HOU} correspond exactly to E^2-unification problem $\langle E, C \rangle$, where

$$C_{E^2} = \{ t_1[\overline{f_n} \mapsto \overline{m_n}[]] \stackrel{?}{=} t_2[\overline{f_n} \mapsto \overline{m_n}[]], \ldots \}$$

$$E = \{ \ldots \vdash \mathsf{APP}(\mathsf{LAM}(x.\, m_1[x]), m_2[]) \equiv m_1[m_2[]] \}$$

E^2-unification algorithm

Input: set R of 2nd-order rewrite rules (directed equalities)
set C of 2nd-order constraints

Output: meta substitution σ

We specify the procedure as a collection of non-deterministic rules

$$\underbrace{c}_{\text{constraint pattern}} \longrightarrow \langle \underbrace{C'}_{\text{new constraints}}, \underbrace{\sigma'}_{\text{new meta substitutions}} \rangle$$

Rules.

① $\forall \overline{x}_n.\ t \overset{?}{=} t \longrightarrow \langle \emptyset, \mathrm{id} \rangle$ **delete**

② $\forall \overline{x}_n.\ F(\overline{t}_p, \overline{x.s_q}) \overset{?}{=} F(\overline{t'}_p, \overline{x.s'_q}) \longrightarrow \langle C, \mathrm{id} \rangle$ **decompose**

$$C := \{ \forall \overline{x}_n.\ t_i \overset{?}{=} t'_i \quad (\text{for all } i = 1, \dots, p),$$
$$\forall \overline{x}_n, x.\ s_j \overset{?}{=} s'_j \quad (\text{for all } j = 1, \dots, q) \}$$

③ $\forall \overline{x}_n.\ m[\overline{t}_k] \overset{?}{=} u \longrightarrow \langle \emptyset, \sigma \rangle$ **eliminate**

$$\sigma := [m[\overline{z}_k] \mapsto u[\overline{t}_k \mapsto \overline{z}_k]]$$

when $\overline{t}_k$ is a list of <u>distinct</u> bound variables (from $\overline{x}_n$)
and m <u>does not</u> occur in u

④ $\forall \overline{x}_n.\ m[\overline{t}_k] \overset{?}{=} F(\overline{u_p}, \overline{x.s_q}) \longrightarrow \langle C, \sigma \rangle$ **imitate**

$$\sigma := [m[\overline{z}_k] \mapsto F(\overline{m_p[\overline{z}_k]}, \overline{x.m'_q[x, \overline{z}_k]})]$$

$$C := \{ \forall \overline{x}_n.\ m_i[\overline{z}_k] \overset{?}{=} u_i \quad (\text{for all } i = 1, \dots, p),$$
$$\forall \overline{x}_n, x.\ m'_j[x, \overline{z}_k] \overset{?}{=} s_j \quad (\text{for all } j = 1, \dots, q) \}$$

when m <u>does not</u> occur in $\overline{u_p}$ and $\overline{s_q}$

$\overline{m_p}$ and $\overline{m'_q}$ are <u>fresh</u> meta variables

⑤ $\forall \overline{x_n}.\ m[\overline{t_k}] \stackrel{?}{=} u \longrightarrow \langle C, \sigma \rangle$ **project**

$$\sigma := [m[\overline{z_k}] \mapsto z_i]$$

$$C := \{ \forall \overline{x_n}.\ t_i \stackrel{?}{=} u \}$$

when m <u>does not</u> occur in u

⑥ $\forall \overline{x_n}.\ F(\overline{t_p}, \overline{x.s_q}) \stackrel{?}{=} u \longrightarrow \langle C, id \rangle$ **mutate**

$$C := \{ \forall \overline{x_n}.\ t_i \stackrel{?}{=} l_i \quad (\text{for all } i = 1, \dots, p),$$
$$\forall \overline{x_n}, x.\ s_j \stackrel{?}{=} l'_j \quad (\text{for all } j = 1, \dots, q),$$
$$\forall \overline{x_n}.\ r \stackrel{?}{=} u \}$$

when $F(\overline{l_p}, \overline{x.l'_q}) \to r$ is in R

⑦ $\forall \overline{x_n}.\ m[\overline{t_k}] \stackrel{?}{=} u \longrightarrow \langle C, \sigma \rangle$ **mutate (meta)**

$$\sigma := [m[\overline{z_k}] \mapsto F(\overline{m_p[\overline{z_k}]}, \overline{x.m'_q[x, \overline{z_k}]})]$$

$$C := \{ \forall \overline{x_n}.\ m_i[\overline{t_k}] \stackrel{?}{=} l_i \quad (\text{for all } i = 1, \dots, p),$$
$$\forall \overline{x_n}, x.\ m'_j[x, \overline{t_k}] \stackrel{?}{=} l'_j \quad (\text{for all } j = 1, \dots, q),$$
$$\forall \overline{x_n}.\ r \stackrel{?}{=} u \}$$

when $F(\overline{l_p}, \overline{x.l'_q}) \to r$ is in R

and m <u>does not</u> occur in u

Claim 2. Proposed E^2-unification procedure is <u>complete</u> in the sense that if a given E^2-unification problem $\langle E, C \rangle$ has at least one solution, then the procedure will stop in finite time and find a solution.

Example 3.

$\boxed{m\ \langle \lambda f.\, f\ y,\ g\rangle \stackrel{?}{=} g\ y}$ $\boxed{m \mapsto \lambda p.(\pi_1 p)(\pi_2 p)}$

① $\mathsf{APP}(m[\,],\ \mathsf{PAIR}(\mathsf{LAM}(f.\ \mathsf{APP}(f, y)),\ g)) \stackrel{?}{=} \mathsf{APP}(g, y)$

mutate $\downarrow$

② $m[\,] \stackrel{?}{=} \mathsf{LAM}(x_1.\ m_1[x_1])$

$\mathsf{PAIR}(\mathsf{LAM}(f.\ \mathsf{APP}(f, y)),\ g) \stackrel{?}{=} m_2[\,]$

$m_1[m_2[\,]] \stackrel{?}{=} \mathsf{APP}(g, y)$

eliminate $\downarrow$ $m[\,] \mapsto \mathsf{LAM}(x_1.\ m_1[x_1])$

eliminate $\downarrow$ $m_2[\,] \mapsto \mathsf{PAIR}(\ldots)$

③ $m_1[\mathsf{PAIR}(\ldots)] \stackrel{?}{=} \mathsf{APP}(g, y)$

mutate (meta) $\downarrow$ $m_1[z_1] \mapsto \mathsf{APP}(m_3[z_1],\ m_4[z_1])$

④ $m_3[\mathsf{PAIR}(\ldots)] \stackrel{?}{=} \mathsf{LAM}(x_2.\ m_5[x_2])$

$m_4[\mathsf{PAIR}(\ldots)] \stackrel{?}{=} m_6[\,]$

$m_5[m_6[\,]] \stackrel{?}{=} \mathsf{APP}(g, y)$

eliminate $\downarrow$ $m_6[\,] \mapsto m_4[\mathsf{PAIR}(\ldots)]$

mutate (meta) $\downarrow$ $m_3[z_1] \mapsto \mathsf{FIRST}(m_7[z_1])$

⑤ $m_7[\mathsf{PAIR}(\ldots)] \stackrel{?}{=} \mathsf{PAIR}(m_8[\,],\ m_9[\,])$

$m_8[\,] \stackrel{?}{=} \mathsf{LAM}(x_2.\ m_5[x_2])$

$m_5[m_4[\mathsf{PAIR}(\ldots)]] \stackrel{?}{=} \mathsf{APP}(g, y)$

project $\downarrow$ $m_7[z_1] \mapsto z_1$

decompose $\downarrow$

$\mathsf{LAM}(f.\ \mathsf{APP}(f, y)) \stackrel{?}{=} m_8[\,]$

$g \stackrel{?}{=} m_9[\,]$

$m_8[\,] \stackrel{?}{=} \mathrm{LAM}(x_2.\, m_5[x_2])$

$m_5[m_4[\mathrm{PAIR}(\dots)]] \stackrel{?}{=} \mathrm{APP}(g, y)$

eliminate $\downarrow$ $m_8[\,] \mapsto \mathrm{LAM}(f.\, \mathrm{APP}(f, y))$

eliminate $\downarrow$ $m_9[\,] \mapsto g$

$\mathrm{LAM}(f.\, \mathrm{APP}(f, y)) \stackrel{?}{=} \mathrm{LAM}(x_2.\, m_5[x_2])$

$m_5[m_4[\mathrm{PAIR}(\dots)]] \stackrel{?}{=} \mathrm{APP}(g, y)$

decompose $\downarrow$

$\forall x_1.\ \mathrm{APP}(x_1, y) \stackrel{?}{=} m_5[x_1]$

$m_5[m_4[\mathrm{PAIR}(\dots)]] \stackrel{?}{=} \mathrm{APP}(g, y)$

eliminate $\downarrow$ $m_5[z_1] \mapsto \mathrm{APP}(z_1, y)$

$\mathrm{APP}(m_4[\mathrm{PAIR}(\dots)], y) \stackrel{?}{=} \mathrm{APP}(g, y)$

decompose $\downarrow$

delete $\downarrow$

$m_4[\mathrm{PAIR}(\dots)] \stackrel{?}{=} g$

mutate (meta) $\downarrow$ $m_4[z_1] \mapsto \mathrm{SECOND}(m_{10}[z_1])$

$m_{10}[\mathrm{PAIR}(\dots)] \stackrel{?}{=} \mathrm{PAIR}(m_{11}[\,], m_{12}[\,])$

$g \stackrel{?}{=} m_{12}[\,]$

project $\downarrow$ $m_{10}[z_1] \mapsto z_1$

decompose $\downarrow$

$\mathrm{LAM}(f.\, \mathrm{APP}(f, y)) \stackrel{?}{=} m_{11}[\,]$

$g \stackrel{?}{=} m_{12}[\,]$

$g \stackrel{?}{=} m_{12}[\,]$

eliminate $\times 3$ $\downarrow$ $m_{11}[\,] \mapsto \mathrm{LAM}(f.\, \mathrm{APP}(f, y))$

$m_{12}[\,] \mapsto g$

$\emptyset$

Current Results

1. Two (equivalent?) versions of the E^2-unification procedure
2. Unproven claims that following HOU problems are special cases
   - HoU for λ-calculus
   - Miller's higher-order pattern unification
3. Unproven claim that the procedure is complete (at least when R is convergent)
4. Unexplored time/space complexity of the procedure
5. Experimental implementation in Haskell fizruk/rzk
   tested with type inference for STLC and MLTT
6. Cleaner implementation to appear at fizruk/e2-unification

Thank You!

∞