E-Unification of Higher-order Patterns

Alexandre Boudet

Evelyne Contejean

LRI Orsay France

Motivations

- Higher-order unification is undecidable (Huet)
- Unification of higher-order patterns is decidable (Miller)

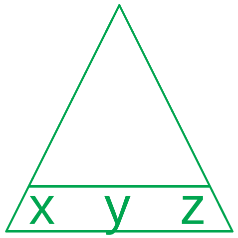
- Combination of algebraic and functional programming paradigms
- Local confluence of HRSs

$$\Downarrow$$

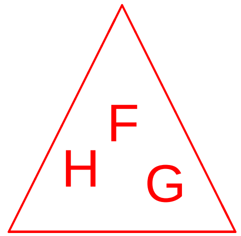
Unification of higher-order patterns
modulo equational theories

Patterns

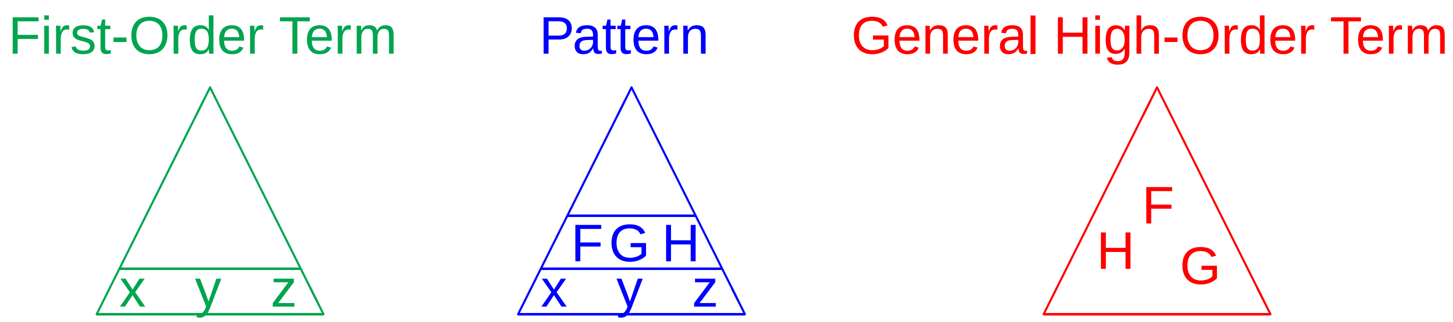
First-Order Term



General High-Order Term



Patterns



Definition Pattern:

- term of the simply-typed λ-calculus in β-normal form
- the arguments of a free variable are η-equivalent to distinct bound variables.

Patterns	Not patterns
$\lambda xyz.f(H(x,y),H(x,z))$	$\lambda xy.G(x,x,y)$
$\lambda x.F(\lambda z.x(z)) =_\eta \lambda x.F(x)$	$\lambda xy.H(F(x),y)$

No equational theory, but α, β, η.

Theorem (Miller)
In the case of patterns, unifiability is decidable
there is an algorithm for computing a mgu.

E-unification of Patterns

Definition

$E = \{l_1 \simeq r_1, \ldots, l_n \simeq r_n\}$: set of First-Order axioms.
Equational theory $=_E$: least congruence containing all the $l_i\sigma \simeq r_i\sigma$
(context, application and abstraction)

Definition

Equation : $s = t$, pair of patterns of the same type.
Unification problem : $\top$, $\bot$ or $P \equiv s_1 = t_1 \ \wedge \ \cdots \ \wedge \ s_n = t_n$.
E-unifier of P : substitution σ such that $\forall i$, $s_i\sigma =_{\beta\eta E} t_i\sigma$.

Theorem (Tannen) $\forall\, u, v \;\; u =_{\beta\eta E} v \Longleftrightarrow u \updownarrow_\beta^\eta =_E v \updownarrow_\beta^\eta$.

How to split when $E = E_1 \cup \ldots \cup E_n$?

Aim : unification of patterns modulo

β, η

$$E_1 \qquad \ldots \qquad E_i \qquad \ldots \qquad E_n$$

How to split when $E = E_1 \cup \ldots \cup E_n$?

Aim : unification of patterns modulo

β, η

$E_1 \quad \ldots \quad E_i \quad \ldots \quad E_n$

Naive approach

How to split when $E = E_1 \cup \ldots \cup E_n$?

Aim : unification of patterns modulo

β, η

$E_1 \quad \ldots \quad E_i \quad \ldots \quad E_n$

Naive approach

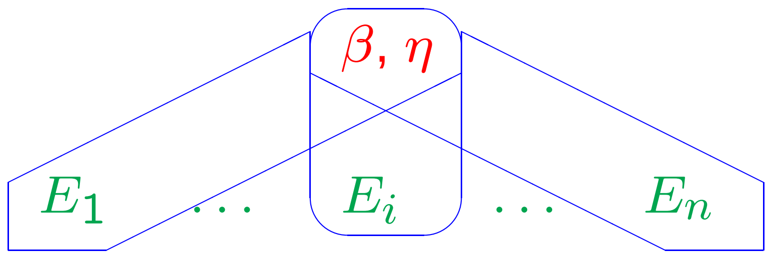
Counter-Example (Qian & Wang) with $E = AC(+)$:

$\lambda xy \cdot F(x, y) = \lambda xy \cdot F(y, x)$ has the solutions

$\forall n \in \mathbb{N} \;\; \sigma_n = \{F \mapsto \lambda xy \cdot G(H_1(x, y) + H_1(y, x), \ldots, H_n(x, y) + H_n(y, x))\}$

How to split when $E = E_1 \cup \ldots \cup E_n$?

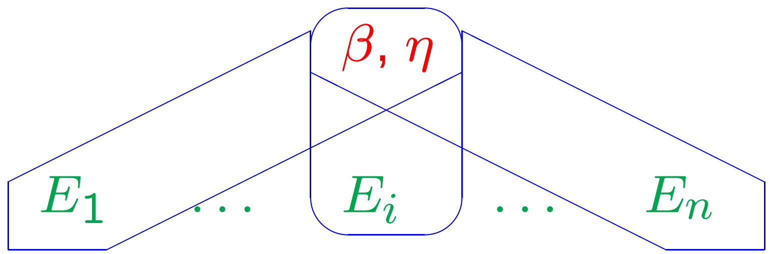
Aim : unification of patterns modulo



Realistic approach

How to split when $E = E_1 \cup \ldots \cup E_n$?

Aim : unification of patterns modulo



Realistic approach

Algorithms for patterns unification modulo the E_is are assumed to be given.

In practice, $\emptyset$, AC, ACU, ACUN, AG

Splitting the unification problem

Definition

Theory of f, algebraic symbol, or of x, bound variable

$Th(f) = E_i$, E_i such that $f \in F_i$ $\quad$ $Th(x) = E_\emptyset$

Alien subterm u in $t[u]_p$: u argument of f and $Th(f) \neq Th(head(u))$.

VA

$\lambda\overline{x}.\mathbf{t}[\mathbf{u}]_\mathbf{p} = \lambda\overline{x}.\mathbf{s} \ \rightarrow \exists\mathbf{H}\ \lambda\overline{x}.\mathbf{t}[\mathbf{H}(\overline{y})]_\mathbf{p} = \lambda\overline{x}.\mathbf{s} \ \wedge\ \lambda\overline{y}.\mathbf{H}(\overline{y}) = \lambda\overline{y}.\mathbf{u}$

if $\mathbf{u}$ is an alien subterm of $\mathbf{t}[\mathbf{u}]_\mathbf{p}$, $\overline{y} = \mathcal{FV}(u) \cap \overline{x}$, and $\mathbf{H}$ new variable.

Split

$$\lambda\overline{x}.\gamma(\overline{s}) = \lambda\overline{x}.\delta(\overline{t}) \ \rightarrow \ \exists\mathbf{F}\ \ \lambda\overline{x}.\mathbf{F}(\overline{x}) = \lambda\overline{x}.\gamma(\overline{s}) \ \wedge \ \lambda\overline{x}.\mathbf{F}(\overline{x}) = \lambda\overline{x}.\delta(\overline{t})$$

if γ and δ not free variables, $Th(\gamma) \neq Th(\delta)$, and $\mathbf{F}$ new variable.

Split unification problem

A unification problem in NF wrt **VA** and **Split**:

$$P \equiv P_F \;\wedge\; P_0 \;\wedge\; P_1 \;\wedge\; \cdots \;\wedge\; P_n$$

- P_F contains all the Flex-Flex equations $\lambda\overline{x}.F(\overline{x}) = \lambda\overline{x}.F(\overline{x}^\pi)$.
- P_0 is pure in E_0, with no $\lambda\overline{x}.F(\overline{x}) = \lambda\overline{x}.F(\overline{x}^\pi)$.
- P_1 is a pure unification problem in E_1.
- P_n is a pure unification problem in E_n.

Notation

$\lambda\overline{x}.F(\overline{x}^\pi)$: $\lambda x_1 \ldots \lambda x_n.F(x_{\pi(1)}, \ldots, x_{\pi(n)})$, where π is a permutation over $\{1, \ldots, n\}$.

A combination algorithm through don't know non-determinism

Guess the actual arguments of a variable

Definition

Constant preserving substitution: $\sigma = \{F \mapsto \lambda\overline{x}.s\}$, $\lambda\overline{x}.s$ in NF and every x_i of $\overline{x}$ has a free occurrence in s.

Projection: $\sigma = \{F \mapsto \lambda\overline{x}.F'(\overline{y}) \mid \{\overline{y}\} \subseteq \{\overline{x}\}\}$

Lemma σ a substitution, then $\sigma \updownarrow_{\beta}^{\eta}= (\pi\theta) \updownarrow_{\beta}^{\eta}$ with π projection and θ constant-preserving substitution.

Project	P	$\rightarrow$	$\exists F'$	$F = \lambda\overline{x}.F'(\overline{y})$ $\wedge$	$P\{F \mapsto \lambda\overline{x}.F'(\overline{y})\}$

where F' is a new variable and $\{\overline{y}\} \subset \{\overline{x}\}$

A combination algorithm through don't know non-determinism

Guess the flex-flex equations

FF$_{\neq}$ $\quad P \quad \rightarrow F = \lambda\overline{x}.G(\overline{x}^{\pi}) \ \wedge \ P\{F \mapsto \lambda\overline{x}.G(\overline{x}^{\pi})\}$

where π is a permutation, types of F and G^{π} are compatible, $F \neq G$ and F and G occur in P.

Guess the permutations over the arguments

FF$_{=}$ $\quad P \quad \rightarrow \lambda\overline{x}.F(\overline{x}) = \lambda\overline{x}.F(\overline{x}^{\pi}) \ \wedge \ P$

where F is a free variable of P, types of F and F^{π} are compatible.

A combination algorithm through don't know non-determinism

Find a representative for each variable

Apply as long as possible

Coalesce

$$\lambda\overline{x}.F(\overline{y}) = \lambda\overline{x}.G(\overline{z}) \ \wedge \ P \quad \rightarrow F = \lambda\overline{y}.G(\overline{z}) \ \wedge \ P\{F \mapsto \lambda\overline{y}.G(\overline{z})\}$$

if $F \neq G$ and $F, G \in \mathcal{FV}(P)$, where $\overline{y}$ is a permutation of $\overline{z}$.

Guess the theory of the representatives

Guess an ordering on representatives

Dealing with $\lambda\overline{x}.F(\overline{x}) = \lambda\overline{x}.F(\overline{x}^\pi)$ by freezing

Example (Qian & Wang) $E = AC(+)$:

$$\lambda xy \cdot F(x,y) = \lambda xy \cdot F(y,x)$$

has the solutions

$$\sigma_n = \{F \mapsto \lambda xy \cdot G(H_1(x,y)+H_1(y,x),\ldots,H_n(x,y)+H_n(y,x))\}$$

for all $n \in \mathbb{N}$.

In addition σ_{n+1} is strictly more general than σ_n (nullary theory).

Solving the pure problems, compatiblity with frozen equations

Definition Solve rule for E_i: algorithm

input: P_i, pure problem in E_i and P_F frozen equations
output: P'_i and P'_{iF} such that

1. $P'_i \equiv \sigma_{E_i}$, is a solved form without flex-flex equations.
2. P'_{iF} is equal to P_F plus some additonnal $\lambda\overline{x}.F(\overline{x}) = \lambda\overline{x}.F(\overline{x}^{\pi})$.
3. F instantiated by σ_{E_i} only if E_i is the chosen theory of F
4. the value of F may contain G only if $F <_{oc} G$, for the chosen ordering
5. for all the equations $s = t$ of P_i and P_F, $s\sigma_{E_i}$ and $t\sigma_{E_i}$ can be proven E_i-equal (by using the equations in P'_{iF}).

Example: AC(+)

Input :

$\lambda xy.F(x,y) + G(x,y) = \lambda xy.2H(x,y) \wedge \lambda xy.H(x,y) = \lambda xy.H(y,x)$

Output :

$$\begin{array}{ll} F = \lambda xy.F'(x,y) + 2F''(x,y) & \\ G = \lambda xy.F'(x,y) + 2F''(y,x) & \wedge\ \lambda xy.F'(x,y) = \lambda xy.F'(y,x) \\ H = \lambda xy.F'(x,y) + F''(x,y) + F''(y,x) & \end{array}$$

In order to prove that $\lambda xy.H\sigma(x,y) =_{AC} \lambda xy.H\sigma(y,x)$, that is

$\lambda xy.F'(x,y) + F''(x,y) + F''(y,x) =_{AC} \lambda xy.F'(y,x) + F''(y,x) + F''(x,y)$,

we need $\lambda xy.F'(x,y) = \lambda xy.F'(y,x)$.

Solving the pure problems, compatiblity with frozen equations

Proposition $s = t$ a pure equation in E_i, and σ such that $s\sigma =_E t\sigma$. Then there exists $P_{perm} = \{\lambda\overline{x}^{\pi}.F(\overline{x}) = \lambda\overline{x}^{\varphi}.F(\overline{x})\}$, σ_{E_i} and θ such that

- $\sigma =_E \sigma_{E_i}\theta$.
- σ_{E_i} pure in E_i,
- θ E-solution of P_{perm}.
- for all pure equations $s' = t'$ (in particular $s = t$) such that $s'\sigma =_E t'\sigma$, $s'\sigma_{E_i}$ and $t'\sigma_{E_i}$ can be proven E_i-equal (using the equations of P_{perm}).

The algorithm

ALGORITHM FOR PATTERN UNIFICATION MODULO $E_0 \cup \cdots \cup E_n$

1. Apply as long as possible the rules **VA** and **Split**.
2. Perform successively the steps of guessing.
3. Apply a **Solve** rule for theory E_i to each P_i.
4. Return $P'_0 \wedge P'_1 \wedge \cdots \wedge P'_n \wedge P_F \wedge \bigwedge_{1 \leq i \leq n} P'_{iF}$.

Main Theorem

Given an equational theory $E = E_0 \cup \cdots \cup E_n$, where the E_is are defined over disjoint signatures $\mathcal{F}_0, \ldots, \mathcal{F}_n$ and a unification problem P, containing only algebraic symbols of $\mathcal{F}_0 \cup \cdots \cup \mathcal{F}_n$,

- The above algorithm returns a constrained DAG-E-solved form of P.
- Every E-unifier of P is a solution of a constrained DAG-solved form computed by the above algorithm.

Theories with a **Solve** rule

- the free theory
- AC, ACU, ACUN, AG
- decomposable syntactic theories, C, DI

Conclusion and perspectives

- more theories (BR?)

- combination of unification algorithms lifted from First-Order terms to Patterns (need of a **Solve** rule for each FO theory for patterns).