Anti-unification: The Other Operation

David M. Cerna

Dynatrace Research, *Czech Academy of Sciences*

June 5^{th} 2025

Unification

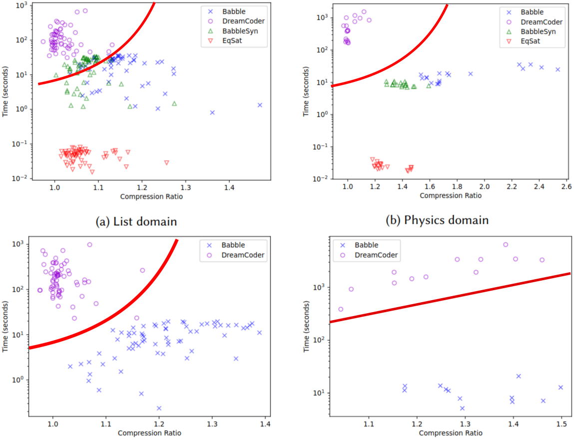
$$s \stackrel{?}{=} t$$

Given two symbolic expressions can we instantiate occuring variables such that the resulting expressions are equivalent?

- Heavily studied subject with multiple comprehensive surveys: (Siekmann, 1989) and (Baader and Synder, 2001)
- Probably another such survey is due.
- **Equating terms** isn't everything!
- Identifying how distinct expressions are related is also fundamental.

The Other Operator

Babble: library learning modulo theory



(a) List domain

(b) Physics domain

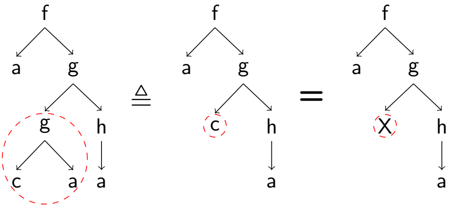
(c) Text domain

(d) Logo domain

Babble: Learning Better Abstractions with E-Graphs and Anti-Unification, Cao *et al.*, POPL 2023

Anti-unification

- Process deriving from a set of symbolic expressions a new expression possessing commonalities shared between its members.



- Independently introduced by Plotkin and Reynolds in 1970.
  - "*A note on inductive generalization*" by G. D. Plotkin
  - "*Transformational systems and the algebraic structure of atomic formulas*" by J.C. Reynolds
- Many applications are covered in the following Survey:

Anti-unification and Generalization: A Survey, D.M. Cerna and T. Kutsia, IJCAI 2023 doi.org/10.24963/ijcai.2023/736

Applications: Inductive Synthesis

- Second-order anti-unification for program Replay.

The Replay of Program Derivations, R.W. Hasker, 1995, Thesis

- θ-subsumption for building bottom clauses.

Inverse entailment and Progol, S. Muggleton, 1995, NGCO

- Lggs used for recursive functional program synthesis.

IGOR II – an Analytical Inductive Functional Programming System, M. Hofmann, 2010, PEPM

- Anti-unification for templating the recursion step.

Inductive Synthesis of Functional Programs: An Explanation Based Generalization Approach, E. Kitzelmann U. Schmid, 2006, JMLR

- Flash-fill in Microsoft Excel.

Programming by Example using Least General Generalizations, By M. Raza, S. Gulwani, N. Milic-Frayling, 2014, AAAI

Applications:Bugs and Optimizations

- Extracting fixes from repository history.

  Learning Quick Fixes from Code Repositories by R. Sousa , *et al.*, 2021, SBES

- Templating bugs with corresponding fixes.

  Getafix: Learning to Fix Bugs Automatically By J. Bader, *et al.*, 2019, OOPSLA

- Templating configuration files to catagorize errors.

  Rex: Preventing Bugs and Misconfiguration in Large Services Using Correlated Change Analysis By Sonu Mehta, *et al.*, 2020, NSDI

- Optimization of recursion schemes for efficient parallelizability.

  Finding parallel functional pearls: Automatic parallel recursion scheme detection in Haskell functions via anti-unification By A. D. Barwell, C. Brown, K. Hammond, 2017, FGCS

Applications: Theorem Proving

- Extraction of substitutions from substitution trees.

  Higher-order term indexing using substitution trees By B. Pientka, 2009, ACM TOCL

- Grammar compression and inductive theorem proving.

  Algorithmic Compression of Finite Tree Languages by Rigid Acyclic Grammars, By S. Eberhard, G. Ebner, S. Hetzl, 2017, ACM TOCL

- Generating SyGuS problems.

  Reinforcement Learning and Data-Generation for Syntax-Guided Synthesis, By J. Parsert and E. Polgreen, 2024, AAAI

Anti-Unification: Basics

- Let Σ be signature, $\mathcal{V}$ a countable set of variables, and $\mathcal{T}(\Sigma, \mathcal{V})$ a term algebra.
- (Anti-Unification) For $s, t \in \mathcal{T}(\Sigma, \emptyset)$:

  Does there exists $g \in \mathcal{T}(\Sigma, \mathcal{V})$ and substitutions σ_s and σ_t s.t. $g\sigma_s = s$ and $g\sigma_t = t$?
- The term g is referred to as a generalization of s and t.
- Observe that $x \in \mathcal{V}$ always generalizes s and t (typically...):

$$\sigma_s = \{x \mapsto s\}\ ,\ \sigma_t = \{x \mapsto t\}$$

- Let's look at an example.

Anti-Unification: Basics

Consider,

$$f(g(b, a)) \triangleq f(g(a, a))$$

- $f(y)$ generalizes the terms,

$$\{y \leftarrow g(b, a)\} \quad \{y \leftarrow g(a, a)\}$$

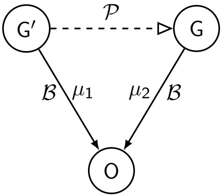
  but, $f(g(y, a))$ is more specific

$$\{y \leftarrow b\} \quad \{y \leftarrow a\}$$

- Let g_1 and g_2 be generalizers of t_1 and t_2, then g_1 is less general then g_2, $g_2 \prec g_1$ if there exists μ s.t. $g_2\mu = g_1$.

- g_1 is least general if for every comparable term g_2, $g_2 \prec g_1$.

A General Framework



Generic	Concrete
$\mathcal{O}$	$\mathcal{T}(\Sigma, \mathcal{V})$
$\mathcal{M}$	First-order substitutions
$\mathcal{B}$	$=$ (syntactic equality)
$\mathcal{P}$	$\preceq$: $s \preceq t$ if $s\sigma = t$ for some σ

- **Goal:** from $O_1, O_2 \in \mathcal{O}$ (symbolic expressions) derive $G \in \mathcal{O}$ possessing certain commonalities shared by O_1 and O_2.
- **Specification:** define (a) a class of mappings $\mathcal{M}$ from $\mathcal{O} \to \mathcal{O}$, (b) a base relation $\mathcal{B}$ consistent with $\mathcal{M}$, and (c) a preference relation $\mathcal{P}$ consistent with $\mathcal{B}$.
- **Result:** G is a $\mathcal{B}$-generalization of O_1 and O_2 and most $\mathcal{P}$-preferred ("better" than G').

A General Framework

- A set $\mathcal{G} \subset \mathcal{O}$ is called $\mathcal{P}$-complete set of $\mathcal{B}$-generalizations of $O_1, O_2 \in \mathcal{O}$ if:
  - **Soundness:** Every $G \in \mathcal{G}$ is a $\mathcal{B}$-generalization of O_1 and O_2.
  - **Completeness:** For each $\mathcal{B}$-generalization G' of O_1 and O_2, there exists $G \in \mathcal{G}$ such that $\mathcal{P}(G', G)$ (G is more preferred).
- Furthermore, $\mathcal{G}$ is minimal if:
  - **Minimality:** No distinct elements of $\mathcal{G}$ are $\mathcal{P}$-comparable: if $G_1, G_2 \in \mathcal{G}$ and $\mathcal{P}(G_1, G_2)$, then $G_1 = G_2$.
- Minimal Complete sets come in four Types:
  - **Unitary** ($\mathbf{1}$): $\mathcal{G}$ is a singleton,
  - **Finitary** (ω): $\mathcal{G}$ is finite and contains at least two elements,
  - **Infinitary** (∞): $\mathcal{G}$ is infinite,
  - **Nullary** ($\mathbf{0}$): $\mathcal{G}$ does not exist (minimality and completeness contradict each other).
- Types are extendable to generalization problems.

Equational Generalization

- Equational considers the same objects ($\mathcal{O}$) as first-order syntactic, but using different base and preference relations:

Generic	Concrete (FOEG)
$\mathcal{O}$	$\mathcal{T}(\Sigma, \mathcal{V})$
$\mathcal{M}$	First-order substitutions
$\mathcal{B}$	$\approx_E$ (equality modulo E)
$\mathcal{P}$	$\preceq_E$ (more specific, less general modulo E) $s \preceq_E t$ iff $s\sigma \approx_E t$ for some σ

Complete sets of solutions

- Here are some examples for each category of complete sets:
  - **UNITARY:**
    - First-Order terms
    - High-Order patterns $\lambda x, y.X(x, y)$
  - **FINITARY:**
    - Permutative theories $f(x, y) = f(y, x)$
    - Unranked Terms and Hedges flexi-arity symbols
    - 1-unital theory $f(e_f, x) = f(x, e_f) = x$
  - **INFINITARY:**
    - Idempotent theories, $f(x, x) = x$
    - Absorptive theories, $f(e_f, x) = f(x, e_f) = e_f$
  - **NULLARY:**
    - 2-Unital Theory $f(e_f, x) = f(x, e_f) = x$
    - Simply typed lambda calculus
    - Cartesian Combinators
    - f(a)=a, f(b)=b
    - TOOOOOO Many

Nullarity Around Every Corner

- Let us focus on the following theory:

$$E = \{f(a) = a, \qquad f(b) = b\}$$

- Why is it **NULLARY**? Consider the problem:

$$a \triangleq b$$

- x is a solution $\{x \mapsto a\}$ and $\{x \mapsto a\}$
- $f^n(x)$, $n \geq 0$ is a solution $\{x \mapsto a\}$ and $\{x \mapsto a\}$
- $x \prec_E f(x) \prec_E f(f(x)) \prec_E \ldots$
- Thus, $mcsg_E(a, b)$ does not exists.
- Many relatively simple theories are Nullary.

Nullarity of 2-unital Theories

- Let us focus on the following theory:

$$E = \{f(x, \epsilon_f) = f(\epsilon_f, x) = x, \qquad g(x, \epsilon_g) = g(\epsilon_g, x) = x\}$$

- To understand the complexity, consider the problem:

$$h(a, a) \triangleq f(h(b, \epsilon_f), b)$$

- x is a solution $\{x \mapsto h(a, a)\}$ and $\{x \mapsto f(h(b, \epsilon_f), b)\}$

- However,
$$h(a, a) \equiv_E f(h(a, a), \epsilon_f).$$
- Thus, $f(h(x, y), z)$ is a solution: $\{x \mapsto a, y \mapsto a, z \mapsto \epsilon_f\}$ and $\{x \mapsto b, y \mapsto \epsilon_f, z \mapsto b\}$
- Observe $f(x, y)$ generalizes $a \triangleq b$.
- Thus, $f(h(f(x, y), x), z)$ is a solution: $\{x \mapsto a, y \mapsto \epsilon_f, z \mapsto \epsilon_f\}$ and $\{x \mapsto \epsilon_f, y \mapsto b, z \mapsto b\}$
- We cannot get more specific using only f.
- What if we use g as well?

Nullarity of 2-unital Theories

- Consider $\epsilon_f \triangleq \epsilon_g$:

$$x \preceq_E f(x, y) \prec_E f(g(x, y), x) \prec_E f(g(x, f(g(x, y), x), x)$$

- Observe x generalizes $\epsilon_f \triangleq \epsilon_g$ and y generalizes $\epsilon_g \triangleq \epsilon_f$
- Generates an infinite sequence of greater specificity.
  - But may contain spiky branches.
  - But may contain mininal generalizations.
- To show: every complete set contains an infinite sequence of greater specificity.

Nullarity of 2-unital Theories

- $\mathbf{g} \in \mathcal{G}_E(s,t)$ and let σ_1 and σ_2 substitutions. Then σ_1 and σ_2 are (s, t, E)-generalizing if
  - $\mathbf{g}\sigma_1 \approx_E s$, $\mathbf{g}\sigma_1 \approx_E t$, and
  - the range of σ_1 and σ_2 are ground.

Definition (Reduced Form)

Let $\mathbf{g} \in \mathcal{G}_E(s,t)$ and let σ_1 and σ_2 be generalizing substitutions. Then $\mathbf{g}$ is in reduced form if

1. For every $x \in var(\mathbf{g})$, $x\sigma_1 \not\approx_{\mathsf{U}} x\sigma_2$, and
2. For $x, y \in var(\mathbf{g})$, $x = y$ or for some $\theta \in \{\sigma_1, \sigma_2\}$, $x\theta \not\approx_{\mathsf{U}} y\theta$.

- Above generalizations are reduced.

Nullarity of 2-unital Theories

Lemma

For every $\mathbf{g} \in \mathcal{G}_E(\epsilon_f, \epsilon_g)$ there exists ϑ such that $\mathbf{g}\vartheta \in \mathcal{G}_E(\epsilon_f, \epsilon_g)$ and reduced.

- Observe:
  1) The set of **reduced** generalizations is complete.
  2) Every **complete** set can be made reduced.

Lemma

Let $\mathbf{g} \in \mathcal{G}_E(\epsilon_f, \epsilon_g)$ be reduced. Then there exists reduced $\mathbf{g}' \in \mathcal{G}_E(\epsilon_f, \epsilon_g)$ such that $\mathbf{g} \prec_E \mathbf{g}'$.

- Reduced generalization are **comparable**.

Nullarity of 2-unital Theories

Theorem

Let $\mathcal{C}$ be a complete set of generalizations of $\epsilon_f \triangleq \epsilon_g$. Then $\mathcal{C}$ contains $\mathbf{g}$ and $\mathbf{g}'$ such that $\mathbf{g} \prec_U \mathbf{g}'$.

Proof.

We can transform $\mathcal{C}$ into a set of reduced generalizations $\mathcal{R}$ which is also complete. We know that for every generalization in $\mathcal{R}$ there exists a more specific generalization. And, because $\mathcal{C}$ is complete, $\mathcal{C}$ must contain an even more specific generalization. □

Corollary

Unital anti-unification is nullary.

Unital anti-unification: Type and algorithms, M. D. Cerna and T. Kutsia, 2020, **Formal Structures, Computation, and Deduction (FSCD)**

Anti-unification over Lambda Terms

- Let $\mathcal{B}$ be a set of base types and Types is the set of types inductively constructed from δ and $\rightarrow$.
- The set Λ is constructed using the following grammar:

$$t ::= x \mid c \mid \lambda x.t \mid t_1 t_2$$

- A lambda term is a pattern if free variables only apply to distinct bound variables.
- $\lambda x.f(X(x), c)$ is a pattern, but $\lambda x.f(X(X(x)), c)$ and $\lambda x.f(X(x, x), c)$ are not.
- Anti-unification of an AUP $X(\vec{x}) : t \triangleq s$ often requires
  - t and s are of the same type ,
  - t and s are in η-long β-normal form,
  - and X does not occur in t and s.

Anti-unification over Lambda Terms

- Calculus of Constructions, pattern fragment.

  Unification and anti-unification in the calculus of construction By F. Pfenning, 1991, LICS

- Anti-unification in $\lambda 2$ ($\mathcal{P}$ based on β-reduction).

  Higher order generalization and its application in program verification, Lu *et al.*, 2000, AMAI

- Pattern Anti-unification in simply-typed λ-calculus.

  Higher-order pattern anti-unification in linear time, A. Baumgartner *et al.*, 2017, JAR

- Top-maximal shallow, simply-typed λ-calculus.

  A generic framework for higher-order generalization, D. Cerna and T. Kutsia, 2019, FSCD

- λ-calculus with recursive let expressions.

  Towards Fast Nominal Anti-unification of Letrec-Expressions, M. Schmidt-Schauß, D. Nantes-Sobrinho *et al.*, 2023, CADE

Rules: Pattern Anti-unification

Dec: **Decomposition**

$\{\underline{X(\vec{x}) : \mathrm{h}(\overline{t_m}) \triangleq \mathrm{h}}(\overline{s_m})\} \uplus A;\ S;\ \sigma \Longrightarrow$
$\{Y_m(\vec{x}) : t_m \triangleq s_m\} \cup A; S;\ G\{X \mapsto \lambda\vec{x}.h(\overline{Y_m(\vec{x})})\},$
where h is constant or $h \in \vec{x}$, and $\overline{Y_m}$ are fresh variables of the appropriate types.

Abs: **Abstraction Rule**

$\{X(\vec{x}) : \lambda y.t \triangleq \lambda z.s\} \uplus A;\ S;\ \sigma \Longrightarrow \{X'(\vec{x}, y) : t \triangleq s\{z \mapsto y\}\} \cup A;\ S;\ G\,\{X \mapsto \lambda\vec{x}, y.X'(\vec{x}, y)\}\,,$
where X' is a fresh variable of the appropriate type.

Extensions: Lambda Terms

Sol: **Solve Rule**

$\{X(\vec{x}) : t \triangleq s\} \uplus A;\ S;\ \sigma \Longrightarrow$
$A;\ \{Y(\vec{y}) : t \triangleq s\} \cup S;\ G\{X \mapsto \lambda\vec{x}.Y(\vec{y})\}$,

where t and s are of a base type, $head(t) \neq head(s)$ or $head(t) = head(s) = \mathrm{h} \notin \vec{x}$. The sequence $\vec{y}$ is a subsequence of $\vec{x}$ consisting of the variables that appear freely in t or in s, and Y is a fresh variable of the appropriate type.

Mer: **Merge Rule**

$A;\ \{X(\vec{x}) : t_1 \triangleq t_2, Y(\vec{y}) : s_1 \triangleq s_2\} \uplus S;\ \sigma \Longrightarrow A;\ \{X(\vec{x}) : t_1 \triangleq t_2\} \cup S;\ G\{Y \mapsto \lambda\vec{y}.X(\vec{x}\pi)\}$,

where $\pi : \{\vec{x}\} \rightarrow \{\vec{y}\}$ is a bijection, extended as a substitution with $t_1\pi = s_1$ and $t_2\pi = s_2$.

Pattern Anti-unification: Example

$\{X : \lambda x, y.f(u(g(x), y), u(g(y), x)) \triangleq \lambda x', y'.f(h(y', g(x')), h(x', g(y')))\};$
$\emptyset; X \Longrightarrow_{\mathsf{Abs}\times 2}$
$\{X'(x, y) : f(u(g(x), y), u(g(y), x)) \triangleq f(h(y, g(x)), h(x, g(y)))\};\ \emptyset;$
$\lambda x, y.X'(x, y) \Longrightarrow_{\mathsf{Dec}}$
$\{Y_1(x, y) : u(g(x), y) \triangleq h(y, g(x)), Y_2(x, y) : u(g(y), x) \triangleq h(x, g(y))\};\ \emptyset;$
$\lambda x, y.f(Y_1(x, y), Y_2(x, y)) \Longrightarrow_{\mathsf{Sol}}$
$\{Y_2(x, y) : u(g(y), x) \triangleq h(x, g(y))\}; \{Y_1(x, y) : u(g(x), y) \triangleq h(y, g(x))\};$
$\lambda x, y.f(Y_1(x, y), Y_2(x, y)) \Longrightarrow_{\mathsf{Sol}}$
$\emptyset;\ \{Y_1(x, y) : u(g(x), y) \triangleq h(y, g(x)), Y_2(x, y) : u(g(y), x) \triangleq h(x, g(y))\};$
$\lambda x, y.f(Y_1(x, y), Y_2(x, y)) \Longrightarrow_{\mathsf{Mer}}$
$\emptyset;\ \{Y_1(x, y) : u(g(x), y) \triangleq h(y, g(x))\};\ \lambda x, y.f(Y_1(x, y), Y_1(y, x))$

Friends of Patterns

- While useful, patterns are quite inexpressive.

> *Functions-as-Constructors Higher-Order Unification*, T. Libal and D. Miller, 2016, FSCD

  - Restricted terms occur as arguments to free variables.
  - Restricted terms are inductively constructed from bound variables and constant symbols with arity > 0.
  - Arguments cannot be subterms of each other.
    - $X(f(x), y)$ is ok, but not $X(f(x), x)$.
  - Arguments cannot be proper subterms of each other.
    - $g(X(f(x), y), Y(f(x), z))$ is ok, but not $g(X(f(x), y), Y(x))$.
- Unitary, but is Finitary without variable restrictions.
- Anti-unification is Unitary without most restrictions.

Friends of Patterns

- Rules construct Top-maximal Shallow Generalizations.
  - $\lambda x.f(X(x))$ is preferred to $\lambda x.X(f(x))$ when possible.
  - $\lambda x.f(X(X(x)))$ or $\lambda x.f(X(Y(x)))$ not allowed.
- Only the Solve rule changes:

Sol: **Solve**

$\{X(\vec{x}) : t \triangleq s\} \uplus A;\ S;\ r \Longrightarrow A;\ \{Y(y_1, \ldots, y_n) :$
$(C_t\, y_1 \cdots y_n) \triangleq (C_s\, y_1 \cdots y_n)\} \cup S;\ r\{X \mapsto \lambda\vec{x}.Y(q_1, \ldots, q_n)\},$

where t and s are of a basic type, $head(t) \neq head(s)$, $q_1, \ldots, q_n$ are distinct subterms of t or s, C_t and C_s are terms such that $(C_t\, q_1 \cdots q_n) = t$ and $(C_s\, q_1 \cdots q_n) = s$, C_t and C_s do not contain any $x \in \vec{x}$, and Y, $y_1, \ldots, y_n$ are distinct fresh variables of the appropriate type.

- Pattern if the $q_1, \ldots, q_n \in \vec{x}$, and $C_t = \lambda\vec{x}.t$ and $C_s = \lambda\vec{x}.s$.

Anti-Unification beyond Patterns

- Not every choice of C_s and C_t will result in a Unitary variant.
- Inconsistent choices for C_s and C_t can result in the computation of non-lggs.
- In particular how the $q_i s$ are chosen matters:
  - $q_i s$ must match a selection condition.
  - $q_i s$ must occur in both terms.
  - $q_i s$ must not be positionally ordered within the terms.
- These conditions allowed us to define 4 Unitary variants.

Anti-Unification beyond Patterns

- Projection Anti-Unification:
  - $q_1 = t,\ q_2 = s,\ C_t = \lambda z_1, z_2.z_1,\ C_s = \lambda z_1, z_2.z_2.$
- Common Subterms Anti-Unification:
  - $q_i s$ position maximal common subterms.
  - $C_t = \lambda y_1, \dots, y_n.\, t[p_1 \mapsto y_1] \cdots [p_m \mapsto y_n]$
  - $C_s = \lambda y_1, \dots, y_n.\, s[l_1 \mapsto y_1] \cdots [l_m \mapsto y_n]$
- Restricted Function-as-constructor Anti-Unification:
  - $q_i s$ position maximal common subterms minus those which break the Local variable condition.
  - C_t and C_s are the same.
- Function-as-constructor Anti-Unification:
  - $q_i s$ position maximal common subterms minus those which break the Local/Global variable conditions.
  - C_t and C_s are the same.
- Other variants are definable (based on the selection condition).

Anti-Unification beyond Patterns: Example

$$\{X : \lambda x.f(h_1(g(g(x)), a, b), h_2(g(g(x)))) \triangleq \lambda y.f(h_3(g(g(y)), g(y), a), h_4(g(g(y))))\};\ \emptyset;\ X$$

$\Longrightarrow_{\text{Abs}}$

$$\{X'(x) : f(h_1(g(g(x)), a, b), h_2(g(g(x)))) \triangleq f(h_3(g(g(x)), g(x), a), h_4(g(g(x))))\}; \emptyset;\ \lambda x.X'(x)$$

$\Longrightarrow_{\text{Dec}}$

$$\{Z_1(x) : h_1(g(g(x)), a, b) \triangleq h_3(g(g(x)), g(x), a), Z_2(x) : h_2(g(g(x))) \triangleq h_4(g(g(x))\};\ \emptyset;$$

$$\lambda x.f(Z_1(x), Z_2(x))$$

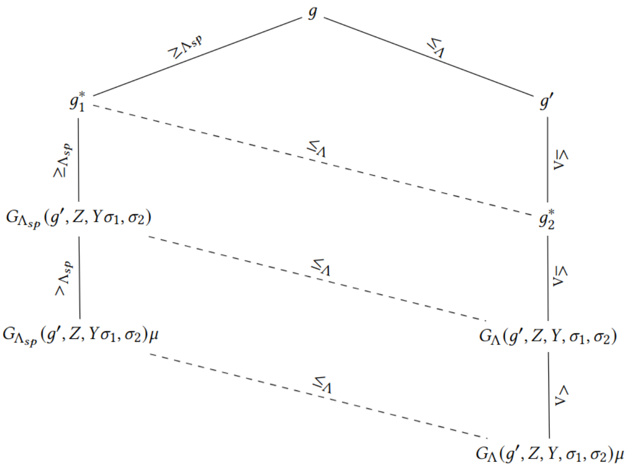
$\Longrightarrow_{\text{Sol-RFC}}$

Anti-Unification beyond Patterns: Example

$$\{Z_2(x) : h_2(g(g(x))) \triangleq h_4(g(g(x))\};$$
$$\{Y_1(y_1) : h_1(g(y_1), a, b) \triangleq h_3(g(y_1), y_1, a)\};$$
$$\lambda x.f(Y_1(g(x)), Z_2(x))$$
$$\Longrightarrow_{\text{Sol-RFC}}$$
$$\emptyset;\ \{Y_1(y_1) : h_1(g(y_1), a, b) \triangleq h_3(g(y_1), y_1, a),$$
$$Y_2(y_2) : h_2(y_2) \triangleq h_4(y_2)\};$$
$$\lambda x.f(Y_1(g(x)), Y_2(g(g(x)))).$$

- Open: Extending this idea to other parts of the lambda Cube, and beyond?
- Beneficial for proof generalization.
- What happens when the terms are no longer shallow?

Deep Lambda Terms: Nullarity



- $\lambda x.\lambda y.f(x) \triangleq \lambda x.\lambda y.f(y)$ has no solution set.
- $\lambda x.\lambda y.f(Z(x,y)) < \lambda x.\lambda y.f(Z(Z(x,y), Z(x,y))) < \cdots$

Deep Lambda Terms: Nullarity

- Its pattern generalization is $g^p = \lambda x.\lambda y.f(Z(x,y))$.
- A generalization more specific g^p is pattern-derived

Definition

Let g be pattern-derived. Then g is *tight* if for all $W \in \mathcal{FV}(g)$:

1) $g\{W \mapsto \lambda \overline{b_k}.b_i\} \notin \mathcal{G}(s,t)$, if W has type $\overline{\gamma_k} \to \gamma_i$ and for $1 \leq i \leq k$ and $\gamma_i \in \mathcal{B}$, and
2) For $(\sigma_1, \sigma_2) \in \mathcal{GS}(s,t,g)$, $g\{W \mapsto t_1\}, g\{W \mapsto t_2\} \notin \mathcal{G}(s,t)$ where $t_1 = W\sigma_1$, $t_2 = W\sigma_2$.

- Observe that **tight** is very similar to **reduced**.

Deep Lambda Terms: Nullarity

Definition

Let $g = \lambda x.\lambda y.f(Z(\overline{s_m}))$ be a tight generalization of $s \triangleq t$ where

1) Z has type $\overline{\delta_m} \to \alpha$ for $1 \leq i \leq m$, and s_i has type δ_i.
2) $(\sigma_1, \sigma_2) \in \mathcal{GS}(s, t, g)$ such that $Z\sigma_1 = r_1$ and $Z\sigma_2 = r_2$,
3) r_1 and r_2 are of type $\overline{\delta_m} \to \alpha$, and
4) Y such that $Y \notin \mathcal{FV}(g)$ and has type $\alpha \to \alpha \to \alpha$.

Then the g-**pseudo-pattern**, denoted $G(g, Z, Y, \sigma_1, \sigma_2)$, is

$$g\{Z \mapsto \lambda\overline{b_m}.Y(r_1(\overline{b_m}), r_2(\overline{b_m}))\} = \lambda x.\lambda y.f(Y(r_1(\overline{q_m}), r_2(\overline{q_m})))$$

where for all $1 \leq i \leq m$, $q_i = s_i\{Z \mapsto \lambda\overline{b_m}.Y(r_1(\overline{b_m}), r_2(\overline{b_m}))\}$.

- Essentially, we regularized the structure of the generalizations.

Deep Lambda Terms: Nullarity

Theorem

For anti-unification of simply-typed lambda terms is nullary.

Proof.

Let us assume that $C \subseteq \mathcal{G}(s, t)$ is minimal and complete. We know C contains a pattern-derived generalization g. Observe that g can be transformed into an tight generalization g' that is also pattern-derived. We can derive a pseudo-pattern generalization g'' of g'. Finally, $g^* = g''\{Y \mapsto \lambda w_1.\lambda w_2.Y(Y(w_1, w_2), Y(w_1, w_2))\}$ is strictly more specific than g''. This implies that $g <_{\mathcal{L}} g^*$, entailing that C is not minimal. □

- Result extendable to non-shallow fragments.

One or nothing: Anti-unification over the simply-typed lambda calculus, D. Cerna and M. Buran, 2024, ACM TOCL.

Equational Anti-unification

- Anti-unification over commutative theories.

  Unification, weak unification, upper bound, lower bound, and generalization problem, F. Baader, 1991, RTA

- Grammar for a **complete** set of E-generalizations:

  E-generalization using grammars, J. Burghardt, 2005, AI

- Minimal complete set of AC-generalizations.

  A modular order-sorted equational generalization algorithm, M. Alpuente *et al.*, 2014, Inf. Comput.

- Minimal complete set of I-generalizations.

  Idempotent anti-unification, D. Cerna and T. Kutsia, 2020, ACM TOCL

- Absorptive Theories.

  Anti-unification over Absorption Theories, M. Ayala-Rincón *et al.*, 2024, IJCAR

E-generalization: Important, but Poorly Behaved...

Unification theory, Jörg H. Siekmann, 1989, Journal Of Symbolic Computation

- Introduces **hierarchy of theories**.
  - Simple theories do not allow subterm collapse
  - {f(a)=a, f(b)=b} is **not** simple.
- Even here strange theories exists:

$$E_1 : \{f(a, g(x)) = f(a, x), f(b, g(x)) = f(b, x)\}$$

$$E_2 : \{s(f(a, g(x))) = f(a, x), s(f(b, g(x))) = f(b, x)\}$$

- E_1 is **Nullary** and E_2 is **Infinitary**.
- Consider the terms $f(a, a)$ and $f(b, b)$.
- Nothing changes if we restrict to **Linear** generalizations:
  - Each variable occurs at most once.

Linearity does matter

- Linear generalizations are easier to construct.
- Applications often use linear rather than non-linear generalizations.
  - well-behaved.
- *Is there a relation between linear and non-linear solutions?*
- consider the following theory:

$$E : \{h(x,b) \approx_E g(x), h(x,a) \approx_E g(x), f(x,g(y)) \approx_E f(x,y)\}$$

- The theory is Simple, but
  - linear is **well-behaved**
  - Non-linear is **Nullary**
- Consider terms $f(a,c)$ and $f(b,d)$.
  - $f(x,y)$
  - $f(x,y)$, $f(x,h(y,x))$, $f(x,h(h(y,x),x))$, ...

Linearity does matter

- For **Ground** Theories such issues only occur if we focus on the problem type.
- Consider the theory:

$$E : \left\{ \begin{array}{c} g(c) = c,\ g(d) = d, \\ f(a,c,d) = e, f(b,c,d) = h, \\ f(a,c,c) = e,\ f(b,d,d) = h \end{array} \right\}$$

- For terms **e** and **h** there is a single linear generalization
- But non-linear generalization is Nullary.
  - $f(x,y,z)$
  - $f(x,g(y),g(y))$, $f(x,g(g(y)),g(g(y)))$,...
- Observe that for **c** and **d**, linear generalization is nullary.

Sufficient Condition

Linear Correspondence

Given a theory E. Does the existence of a minimal complete set of linear generalizations imply the existence of a minimal complete set of generalizations?

- For **Ground** Theories linear correspondance holds.

Lemma

Let E be a ground equational theory, $s, t \in \mathcal{T}(\Sigma, \emptyset)$, and $p \in pos(s)$ such that $s|_p \in \mathcal{V}$. Then if $s \approx_E t$, then $p \in pos(t)$ and $path(s, p) = path(t, p)$.

- Ground theories cannot apply to non-ground terms.

Beyond Ground Theories

- Consider the following theory from

Anti-unification over Absorption Theories, M. Ayala-Rincón *et al.*, 2024, IJCAR

$$E : \{f(\epsilon_f, x) = \epsilon_f,\ f(x, \epsilon_f) = \epsilon_f\}$$

- Linear generalization is **Finitary**, Non-linear is **Infinitary**
- We refer to such theories as Semi-ground.
  - One side is **ground**, One side is **not**.
- **Hard** to define!
- Question: Linear Correspondance??
- Question: More general theories with Linear Correspondance??

Selection Heuristics

- How to deal with the explosion?
  - Alignment and Rigidity functions
  - Skeletons
  - beam search
  - Syntactic restriction
- **Recent Direction:**
  - Should the preference and base relations be Crisp?
  - Are most lggs too distant from the generalized terms to be generalizations?
- Is **similarity** and **quantitative** anti-unification a fix?

A Framework for Approximate Generalization in Quantitative Theories, T. Kutsia and C. Pau, 2022, FSCD

Future Work

- Investigating the above questions
- New applications for anti-unification
- Developing methods for combining anti-unification algorithms for disjoint equational theories

  Combining Generalization Algorithms in Regular Collapse-Free Theories, M. Ayala-Rincón, D. Cerna, T. Kutsia and C. Ringeissen, 2025, FSCD
- Studying computational complexity and optimizations.
- risc.jku.at/sw/unification-and-anti-unification-algorithm-library/