Verifying Nominal Equational Reasoning Modulo Algorithms

The library https://github.com/nasa/pvslib/nominal

Mauricio Ayala-Rincón

2nd Workshop on the development, maintenance, refactoring and search of large libraries of proofs
Tbilisi, 13th September 2024

Mathematics and Computer Science Departments

Universidade de Brasília

† Research supported by the Brazilian agencies CAPES, CNPq, and FAPDF

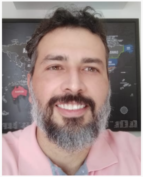
Joint Work With

Ana R. Oliveira

María Júlia Lima

Maribel Fernández

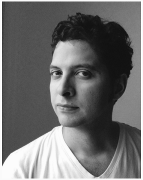
Daniele Nantes



Washington Ribeiro

Gabriela Ferreira

Thaynara de Lima



Mariano Moscato

Gabriel Silva

Andrés González

David Cerna

Temur Kutsia

Outline

Motivation

Equational Problems

- **Equality check**: $s = t$?
- **Matching**: There exists σ such that $s\sigma = t$?
- **Unification**: There exists σ such that $s\sigma = t\sigma$?
- **Anti-unification**: There exist r, σ and ρ such that $r\sigma = s$ and $r\rho = t$?

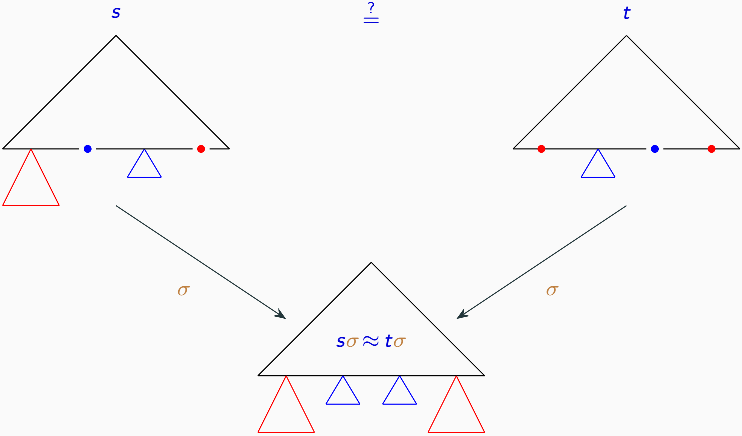
s and t, and u are *terms* in some *signature* and σ and ρ are *substitutions*.

Motivation

Unification modulo

Unification

Goal: find a substitution that identifies two expressions.



Syntactic Unification

- Goal: *to identify* two expressions.
- Method: replace variables by other expressions.

Example: for x and y variables, a and b constants, and f a function symbol,

- *Identify* $f(x, a)$ and $f(b, y)$

Syntactic Unification

- Goal: *to identify* two expressions.
- Method: replace variables by other expressions.

Example: for x and y variables, a and b constants, and f a function symbol,

- *Identify* $f(x, a)$ and $f(b, y)$
- solution $\{x/b, y/a\}$.

Example:

- Solution $\sigma = \{x/b\}$ for $f(x, y) = f(b, y)$ is *more general* than solution $\gamma = \{x/b, y/b\}$.

σ is *more general* than γ:

$$\text{there exists } \delta \text{ such that } \sigma\delta = \gamma;$$

$$\delta = \{y/b\}.$$

Syntactic Unification

Interesting questions:

- Decidability, Unification Type, Correctness and Completeness.
- Complexity.
- With adequate data structures, there are linear solutions (Martelli-Montanari 1976, Petterson-Wegman 1978).

Syntactic unification is of type *unary* and linear.

Unification Modulo

When operators have algebraic equational properties, the problem is not as simple.

Example: for f *commutative* (C), $f(x,y) \approx f(y,x)$:

- $f(x,y) = f(a,b)$?

The unification problem is of type *finitary*.

Unification Modulo

When operators have algebraic equational properties, the problem is not as simple.

Example: for f *commutative* (C), $f(x, y) \approx f(y, x)$:

- $f(x, y) = f(a, b)$?
- Solutions: $\{x/a, y/b\}$ and $\{x/b, y/a\}$.

The unification problem is of type *finitary*.

Example: for f *associative* (A), $f(f(x,y),z) \approx f(x,f(y,z))$:

- $f(x,a) = f(a,x)$?

The unification problem is of type *infinitary*.

Unification Modulo

Example: for f *associative* (A), $f(f(x,y),z) \approx f(x,f(y,z))$:

- $f(x,a) = f(a,x)$?
- Solutions: $\{x/a\}$, $\{x/f(a,a)\}$, $\{x/f(a,f(a,a))\}, \ldots$

The unification problem is of type *infinitary*.

Unification Modulo

Example: for f AC with *unity* (U), $f(x,e) \approx x$:

- $f(x,y) = f(a,b)$?

The unification problem is of type *finitary*.

Example: for f AC with *unity* (U), $f(x,e) \approx x$:

- $f(x,y) = f(a,b)$?
- Solutions: $\{x/e, y/f(a,b)\}$, $\{x/f(a,b), y/e\}$, $\{x/a, y/b\}$, and $\{x/b, y/a\}$.

The unification problem is of type *finitary*.

Unification Modulo

Example: for f A, and *idempotent* (I), $f(x,x) \approx x$:

- $f(x, f(y,x)) = f(f(x,z),x)$?

The unification problem is of type *zero* (Schmidt-Schauß 1986, Baader 1986).

Unification Modulo

Example: for f A, and *idempotent* (I), $f(x,x) \approx x$:

- $f(x, f(y,x)) = f(f(x,z),x)$?
- Solutions: $\{y/f(u, f(x,u)), z/u\}, \dots$

The unification problem is of type *zero* (Schmidt-Schauß 1986, Baader 1986).

Unification Modulo

Example: for $+$ AC, and h *homomorphism* (h), $h(x+y) \approx h(x)+h(y)$:

- $h(y)+a = y+z$?

The unification problem is of type *zero* and undecidable (Narendran 1996). The same happens for ACUh (Nutt 1990, Baader 1993).

Unification Modulo

Example: for $+$ AC, and h *homomorphism* (h), $h(x+y) \approx h(x)+h(y)$:

- $h(y)+a = y+z$?
- Solutions: $\{y/a, z/h(a)\}, \{y/h(a)+a, z/h^2(a)\}, \ldots,$ $\{y/h^k(a)+\ldots+h(a)+a, z/h^{k+1}(a)\}, \ldots$

The unification problem is of type *zero* and undecidable (Narendran 1996). The same happens for ACUh (Nutt 1990, Baader 1993).

		Synthesis Unification modulo			
Theory	Unif. type	Equality-checking	Matching	Unification	Related work
Syntactic	1	O(n)	O(n)	O(n)	R65 MM76 PW78
C	ω	O(n^2)	NP-comp.	NP-comp.	BKN87 KN87
A	∞	O(n)	NP-comp.	NP-hard	M77 BKN87
AU	∞	O(n)	NP-comp.	decidable	M77 KN87
AI	0	O(n)	NP-comp.	NP-comp.	Klíma02 SS86 Baader86

Synthesis Unification modulo

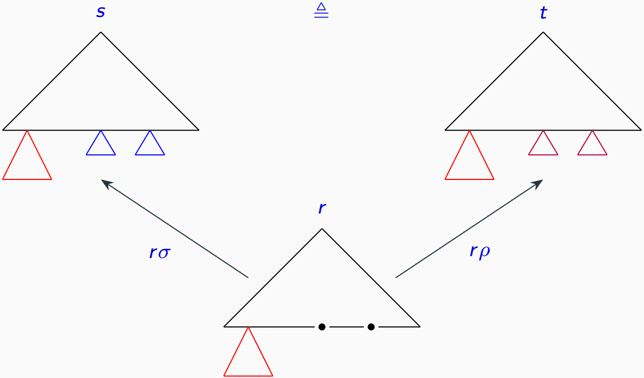
		Synthesis Unification modulo			
Theory	Unif. type	Equality-checking	Matching	Unification	Related work
AC	ω	$O(n^3)$	NP-comp.	NP-comp.	BKN87 KN87 KN92
ACU	ω	$O(n^3)$	NP-comp.	NP-comp.	KN92
AC(U)I	ω	-	-	NP-comp.	KN92 BMMO20
D	ω	-	NP-hard	NP-hard	TA87
ACh	0	-	-	undecidable	B93 N96 EL18
ACUh	0	-	-	undecidable	B93 N96

Motivation

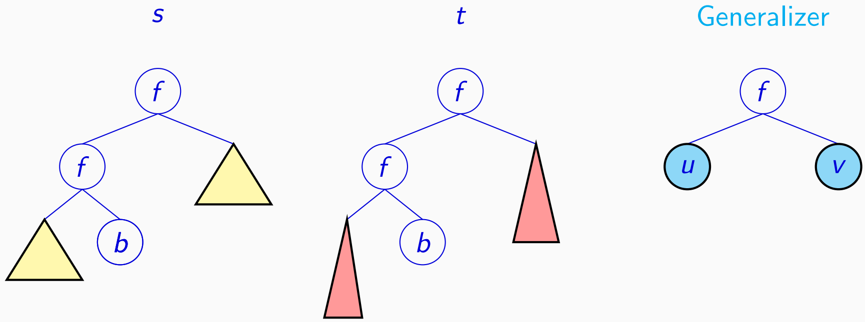
Anti-unification

Anti-unification

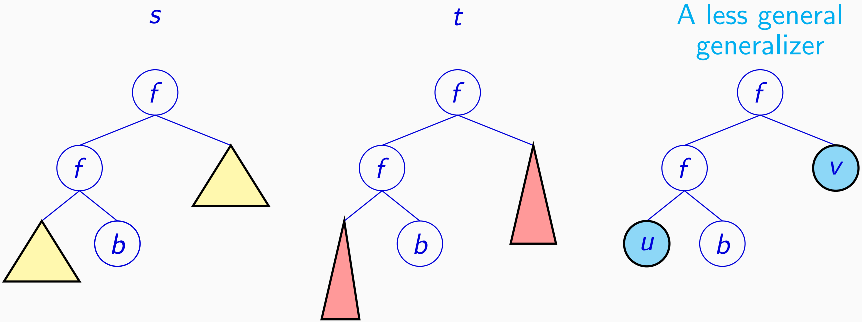
Goal: find the commonalities between two expressions.



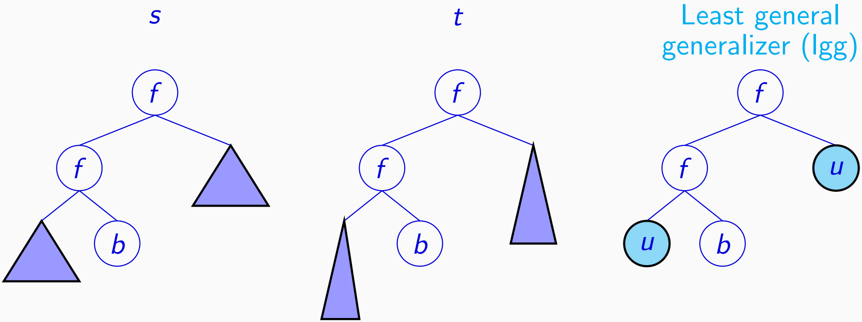
Anti-Unification



Anti-Unification



Anti-Unification



Anti-unification - History

- Introduced by Gordon Plotkin [Plo70] and John Reynolds [Rey70]
- First-order: syntactic [Baa91]; C, A, and AC [AEEM14]; idempotent [CK20b], unital [CK20c], semirings [Cer20], absorption [ACBK24]
- Higher-Order: patterns [BKLV17], top maximal and shallow generalizations variants [CK20a], equational patterns [CK19], modulo [CK20a]
- See david Cerna and Temur Kutsia survey [CK23].

Motivation

Syntactic anti-unification

- terms $t ::= x \mid \langle\rangle \mid \langle t, t\rangle \mid f\ t$
- Labelled equations $E = \{s_i \underset{x_i}{\triangleq} t_i \mid i \leq n\}$

Configurations: $\left\langle \underbrace{E_U}_{\text{Unsolved equations}} \quad \underbrace{E_S}_{\text{Solved equations}} \quad \underbrace{\sigma}_{\text{Substitution}} \right\rangle$

Configuration constraints

- All labels in $E_U \cup E_S$ are different,
- no *redundant* equations appear in E_S, and
- no label in $E_U \cup E_S$ belongs to $dom(\sigma)$.

Inference Rules

$$\text{(Decompose Function)}\ \frac{\langle \{f\ s \overset{\triangle}{\underset{x}{=}} f\ t\} \cup E, S, \sigma \rangle}{\langle \{s \overset{\triangle}{\underset{y}{=}} t\} \cup E, S, \{x \mapsto f\ y\} \circ \sigma \rangle}$$

$$\text{(Decompose Pair)}\ \frac{\langle \{\langle s, u \rangle \overset{\triangle}{\underset{x}{=}} \langle t, v \rangle\} \cup E, S, \sigma \rangle}{\langle \{s \overset{\triangle}{\underset{y}{=}} t, u \overset{\triangle}{\underset{z}{=}} v\} \cup E, S, \{x \mapsto \langle y, z \rangle\} \circ \sigma \rangle}$$

$$\text{(Solve-Red)}\ \frac{\langle \{s \overset{\triangle}{\underset{x}{=}} t\} \cup E, S, \sigma \rangle}{\langle E, S, \{x \mapsto x'\} \circ \sigma \rangle}\ \text{if } s \overset{\triangle}{\underset{x'}{=}} t \in S$$

$$\text{(Solve-No-Red)}\ \frac{\langle \{s \overset{\triangle}{\underset{x}{=}} t\} \cup E, S, \sigma \rangle}{\langle E, \{s \overset{\triangle}{\underset{x}{=}} t\} \cup S, \sigma \rangle}\ \text{if there is no } s \overset{\triangle}{\underset{x'}{=}} t \in S$$

$$\text{(Syntactic)}\ \frac{\langle \{s \overset{\triangle}{\underset{x}{=}} s\} \cup E, S, \sigma \rangle}{\langle E, S, \{x \mapsto s\} \circ \sigma \rangle}\ \text{if neither decomposable nor solvable}$$

Example

$$
\begin{array}{ll}
& \langle \{ f\langle f\langle c,b\rangle ,c\rangle \triangleq_{x} f\langle f\langle d,b\rangle ,d\rangle \},\emptyset, id\rangle \\
\text{(DecFun)} & \overline{\langle \{ \langle f\langle c,b\rangle ,c\rangle \triangleq_{y} \langle f\langle d,b\rangle ,d\rangle \},\emptyset,\{x\mapsto f\ y\}\rangle} \\
\text{(DecPair)} & \overline{\langle \{ f\langle c,b\rangle \triangleq_{z_1} f\langle d,b\rangle ,c\triangleq_{z_2} d\},\emptyset,\{x\mapsto f\ \langle z_1,z_2\rangle \}\rangle} \\
\text{(DecFun)} & \overline{\langle \{ \langle c,b\rangle \triangleq_{z_3} \langle d,b\rangle ,c\triangleq_{z_2} d\},\emptyset,\{x\mapsto f\ \langle f\ z_3,z_2\rangle \}\rangle} \\
\text{(DecPair)} & \overline{\langle \{ c\triangleq_{z} d, b\triangleq_{z_4} b, c\triangleq_{z_2} d\},\emptyset,\{x\mapsto f\ \langle f\ \langle z,z_4\rangle ,z_2\rangle \}\rangle} \\
\text{(SolveNRed)} & \overline{\langle \{ b\triangleq_{z_4} b, c\triangleq_{z_2} d\},\{c\triangleq_{z} d\},\{x\mapsto f\ \langle f\ \langle z,z_4\rangle ,z_2\rangle \}\rangle} \\
\text{(Syntactic)} & \overline{\langle \{ c\triangleq_{z_2} d\},\{c\triangleq_{z} d\},\{x\mapsto f\ \langle f\ \langle z,b\rangle ,z_2\rangle \}\rangle} \\
\text{(SolRed)} & \overline{\emptyset,\{c\triangleq_{z} d\},\{x\mapsto f\ \langle f\ \langle z,b\rangle ,z\rangle \}\rangle}
\end{array}
$$

Motivation

Anti-unification modulo

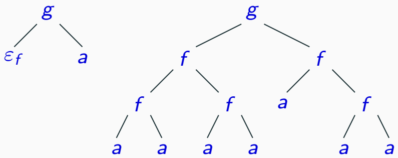
Anti-unification modulo

- Interest on the formalization of anti-unification for theories with Commutative, Associative and Absorption-symbols: C-, A-, and $\mathfrak{a}$-symbols.
- Related $\mathfrak{a}$-symbols are a pair of a function and a constant symbol holding the axioms $f(\varepsilon_f, x) = \varepsilon_f = f(x, \varepsilon_f)$.

Anti-unification in $(\mathfrak{a})(A)(C)(\mathfrak{a}A)(\mathfrak{a}C)$-theories

Example

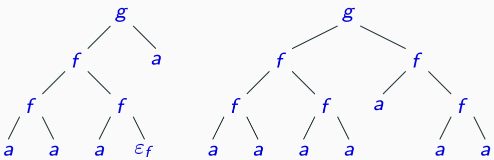
Consider the terms:



An $\mathfrak{a}$-generalization and $\mathfrak{a}$A-generalization will be illustrated.

Anti-unification in ($\mathfrak{a}$)(A)(C)($\mathfrak{a}$A)($\mathfrak{a}$C)-theories

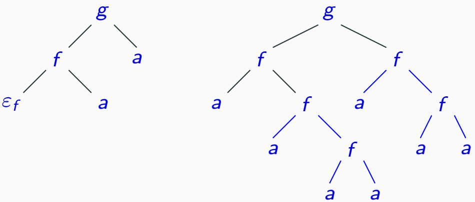
By *expanding* ε_f in $g(\varepsilon_f, a)$, one obtains:



Notice that $g(f(f(a, a), f(a, x)), y)$ is an $\mathfrak{a}$-generalization.

Anti-unification in ($\mathfrak{a}$)(A)(C)($\mathfrak{a}$A)($\mathfrak{a}$C)-theories

Considering the same terms modulo $\mathfrak{a}$A, and by *expanding* ε_f in $g(\varepsilon_f, a)$, one has:



$g(f(x, y), y)$ is an $\mathfrak{a}$A-generalization but not an $\mathfrak{a}$-generalization.

Anti-unification modulo types

Theory	Anti-unification type	References
Syntactic	1	[Plo70, Rey70]
A	ω	[AEEM14]
C	ω	[AEEM14]
† $(U)^1$	ω	[CK20c]
$(U)^{\geq 2}$	nullary	[CK20c]
‡ $\mathfrak{a}$	∞	[ACBK24]
$\mathfrak{a}(C)$	∞	[ACBK24]

$(\dagger)$Unital: $\{f(i_f, x) = f(x, i_f) = x\}$

$(\ddagger)$Absorption $f(\varepsilon_f, x) = \varepsilon_f = f(x, \varepsilon_f)$

Bindings and Nominal Syntax

Systems with Bindings

Systems with bindings frequently appear in mathematics and computer science but are not captured adequately in first-order syntax.

For instance, the formulas

$$\forall x_1, x_2 : x_1 + 1 + x_2 > 0 \quad \text{and} \quad \forall y_1, y_2 : 1 + y_2 + y_1 > 0$$

are not syntactically equal but should be considered equivalent in a system with binding and AC operators.

Nominal

The nominal setting extends first-order syntax, replacing the concept of syntactical equality with α-equivalence, letting us represent those systems smoothly.

Profiting from the nominal paradigm implies adapting basic notions (substitution, rewriting, equality) to it.

Atoms and Variables

Consider a set of variables $\mathbb{X} = \{X, Y, Z, \ldots\}$ and a set of atoms $\mathbb{A} = \{a, b, c, \ldots\}$.

Definition 1 (Nominal Terms)
Nominal terms are inductively generated according to the grammar:

$$s, t \quad ::= \quad a \mid \pi \cdot X \mid \langle\rangle \mid [a]t \mid \langle s, t\rangle \mid f\ t \mid f^{AC}\, t$$

where π is a permutation that exchanges a finite number of atoms.

Permutations

An atom permutation π represents an exchange of a finite amount of atoms in $\mathbb{A}$ and is presented by a list of swappings:

$$\pi = (a_1\ b_1) :: ... :: (a_n\ b_n) :: nil$$

Examples of Permutation Actions

Permutations act on atoms and terms:

- $(a\ b) \cdot a = b$;
- $(a\ b) \cdot b = a$;
- $(a\ b) \cdot f(a, c) = f(b\ c)$;
- $(a\ b) :: (b\ c) \cdot [a]\langle a, c\rangle = (b\ c)[b]\langle b, c\rangle = [c]\langle c, b\rangle$.

Intuition Behind the Concepts

Two important predicates are the *freshness* predicate $\#$, and the α-*equality* predicate $\approx_\alpha$.

- $a \# t$ means that if a occurs in t then it must do so under an abstractor $[a]$.
- $s \approx_\alpha t$ means that s and t are α-equivalent.

Contexts

A *context* is a set of constraints of the form $a\#X$. Contexts are denoted by the letters Δ, ∇ or Γ.

Advantages of the name binding nominal approach

- First-order terms with binders and *implicit* atom dependencies.
- Easy syntax to present *name binding* predicates as $a \in \mathit{FreeVar}(M)$, $a \in \mathit{BoundVar}([a]s)$, and operators as renaming: $(a\ b) \cdot s$.
- Built-in α-equivalence and first-order *implicit substitution*.
- Feasible syntactic equational reasoning: efficient equality-check, matching, and unification algorithms.

Derivation Rules for Freshness

$$\frac{}{\Delta \vdash a\#\langle\rangle}\ (\#\langle\rangle) \qquad \frac{}{\Delta \vdash a\#b}\ (\#atom)$$

$$\frac{(\pi^{-1}(a)\#X) \in \Delta}{\Delta \vdash a\#\pi \cdot X}\ (\#X) \qquad \frac{}{\Delta \vdash a\#[a]t}\ (\#[a]a)$$

$$\frac{\Delta \vdash a\#t}{\Delta \vdash a\#[b]t}\ (\#[a]b) \qquad \frac{\Delta \vdash a\#s \quad \Delta \vdash a\#t}{\Delta \vdash a\#\langle s, t\rangle}\ (\#pair)$$

$$\frac{\Delta \vdash a\#t}{\Delta \vdash a\#f\ \ t}\ (\#app)$$

Derivation Rules for alpha-Equivalence

$$\frac{}{\Delta \vdash \langle\rangle \approx_\alpha \langle\rangle}\ (\approx_\alpha \langle\rangle) \qquad \frac{}{\Delta \vdash a \approx_\alpha a}\ (\approx_\alpha atom)$$

$$\frac{\Delta \vdash s \approx_\alpha t}{\Delta \vdash fs \approx_\alpha ft}\ (\approx_\alpha app) \qquad \frac{\Delta \vdash s \approx_\alpha t}{\Delta \vdash [a]s \approx_\alpha [a]t}\ (\approx_\alpha [a]a)$$

$$\frac{\Delta \vdash s \approx_\alpha (a\ b)\cdot t,\ a\#t}{\Delta \vdash [a]s \approx_\alpha [b]t}\ (\approx_\alpha [a]b) \qquad \frac{ds(\pi,\pi')\#X \subseteq \Delta}{\Delta \vdash \pi \cdot X \approx_\alpha \pi' \cdot X}\ (\approx_\alpha var)$$

$$\frac{\Delta \vdash s_0 \approx_\alpha t_0,\ \ \Delta \vdash s_1 \approx_\alpha t_1}{\Delta \vdash \langle s_0, s_1\rangle \approx_\alpha \langle t_0, t_1\rangle}\ (\approx_\alpha pair)$$

Additional Rule for alpha-Equivalence with C Functions

Let f be a C function symbol.

We add rule ($\approx_\alpha$ *c-app*) for dealing with C functions:

$$\frac{\Delta \vdash s_2 \approx_\alpha t_1 \qquad \Delta \vdash s_1 \approx_\alpha t_2}{\Delta \vdash f^C\langle s_1, s_2\rangle \approx_\alpha f^C\langle t_1, t_2\rangle}$$

Additional Rule for alpha-Equivalence with AC Functions

Let f be an AC function symbol.

We add rule $(\approx_\alpha \textit{ ac-app})$ for dealing with AC functions:

$$\frac{\Delta \vdash S_i(f^{AC} s) \approx_\alpha S_j(f^{AC} t) \quad \Delta \vdash D_i(f^{AC} s) \approx_\alpha D_j(f^{AC} t)}{\Delta \vdash f^{AC} s \approx_\alpha f^{AC} t}$$

$S_n(f*)$ selects the n^{th} argument of the *flattened* subterm $f*$.
$D_n(f*)$ deletes the n^{th} argument of the *flattened* subterm $f*$.

Deriving $\vdash \forall[a] \oplus \langle a, fa\rangle \approx_\alpha \forall[b] \oplus \langle fb, b\rangle$, where $\oplus$ is C:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{a \approx_\alpha a}\,(\approx_\alpha \mathit{atom}) \qquad \dfrac{\dfrac{}{a \approx_\alpha a}\,(\approx_\alpha \mathit{atom})}{fa \approx_\alpha fa}\,(\approx_\alpha \mathit{app})}{\oplus\langle a, fa\rangle \approx_\alpha (a\ \ b) \cdot \oplus\langle fb, b\rangle}\,(\approx_\alpha \mathit{c\text{-}app}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathit{atom})}{a\#fb}\,(\#\mathit{app}) \qquad \dfrac{}{a\#b}\,(\#\mathit{atom})}{a\#\langle fb, b\rangle}\,(\#\mathit{pair})}{a\# \oplus \langle fb, b\rangle}\,(\#\mathit{app})}{[a] \oplus \langle a, fa\rangle \approx_\alpha [b] \oplus \langle fb, b\rangle}\,(\approx_\alpha [a]b)}{\forall[a] \oplus \langle a, fa\rangle \approx_\alpha \forall[b] \oplus \langle fb, b\rangle}\,(\approx_\alpha \mathit{app})
$$

Nominal C-unification

Nominal C-unification

Unification problem: $\langle \Gamma, \{s_1 \approx_\alpha{}^? t_1, \dots s_n \approx_\alpha{}^? t_n\}\rangle$

Unification solution: $\langle \Delta, \sigma\rangle$, such that

- $\Delta \vdash \Gamma\sigma$;
- $\Delta \vdash s_i\sigma \approx_\alpha t_i\sigma, 1 \leq i \leq n$.

We introduced nominal (equality-check, matching) and unification algorithms that provide solutions given as triples of the form:

$$\langle \Delta, \sigma, FP\rangle$$

where FP is a set of fixed-point equations of the form $\pi \cdot X \approx_\alpha{}^? X$.

This provides a finite representation of the infinite set of solutions that may be generated from such fixed-point equations.

Nominal C-unification

Fixed point equations such as $\pi \cdot X \approx_\alpha{}^? X$ may have infinite independent solutions.

For instance, in a signature in which $\oplus$ and $\star$ are C, the unification problem: $\langle \emptyset, \{(a\ b)X \approx_\alpha{}^? X\}\rangle$

$$\text{has solutions: } \begin{cases} \langle \{a\#X, b\#X\}, id\rangle, \\ \langle \emptyset, \{X/a \oplus b\}\rangle, \langle \emptyset, \{X/a \star b\}\rangle, \ldots \\ \langle \{a\#Z, b\#Z\}, \{X/(a \oplus b) \oplus Z\}\rangle, \ldots \\ \langle \emptyset, \{X/(a \oplus b) \star (b \oplus a)\}\rangle, \ldots \end{cases}$$

Issues Adapting First-Order to Nominal AC-Unification

Our Work in First-Order AC-Unification in a Nutshell

We modified Stickel-Fages's seminal AC-unification algorithm to avoid mutual recursion and verified it in the PVS proof assistant.

We formalised the algorithm's termination, soundness, and completeness [AFSS22].

An Example

Let f be an AC function symbol. The solutions that come to mind when unifying:

$$f(X, Y) \approx^? f(a, W)$$

are:

$$\{X \to a, Y \to W\} \text{ and } \{X \to W, Y \to a\}$$

Are there other solutions?

An Example

Yes!

For instance, $\{X \to f(a, Z_1),\ Y \to Z_2,\ W \to f(Z_1, Z_2)\}$ and $\{X \to Z_1,\ Y \to f(a, Z_2),\ W \to f(Z_1, Z_2)\}$.

Example

the **AC Step** for AC-unification.

How do we generate a complete set of unifiers for:

$$f(X, X, Y, a, b, c) \approx^{?} f(b, b, b, c, Z)$$

Stickel-Fages AC-unification - eliminating Common Arguments

Eliminate common arguments in the terms we are trying to unify.

Now, we must unify

$$f(X, X, Y, a) \approx^? f(b, b, Z)$$

Stickel-Fages AC-unification - introducing a Linear equation

According to the number of times each argument appears, transform the unification problem into a linear equation on $\mathbb{N}$:

$$2X_1 + X_2 + X_3 = 2Y_1 + Y_2,$$

Above, variable X_1 corresponds to argument X, variable X_2 corresponds to argument Y, and so on.

Stickel-Fages AC-unification - building a basis of solutions

Generate a basis of solutions to the linear equation.

Table 1: Solutions for the Equation $2X_1 + X_2 + X_3 = 2Y_1 + Y_2$

X_1	X_2	X_3	Y_1	Y_2	$2X_1 + X_2 + X_3$	$2Y_1 + Y_2$
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	0	2	1	0	2	2
0	1	1	1	0	2	2
0	2	0	1	0	2	2
1	0	0	0	2	2	2
1	0	0	1	0	2	2

Stickel-Fages AC-unification - associating new variables

Associate new variables with each solution.

Table 2: Solutions for the Equation $2X_1 + X_2 + X_3 = 2Y_1 + Y_2$

X_1	X_2	X_3	Y_1	Y_2	$2X_1 + X_2 + X_3$	$2Y_1 + Y_2$	**New Variables**
0	0	1	0	1	1	1	Z_1
0	1	0	0	1	1	1	Z_2
0	0	2	1	0	2	2	Z_3
0	1	1	1	0	2	2	Z_4
0	2	0	1	0	2	2	Z_5
1	0	0	0	2	2	2	Z_6
1	0	0	1	0	2	2	Z_7

Stickel-Fages AC-unification - old and new variables

Observing the previous Table, relate the "old" variables and the "new" ones:

$$
\begin{aligned}
X_1 &\approx^? Z_6 + Z_7 \\
X_2 &\approx^? Z_2 + Z_4 + 2Z_5 \\
X_3 &\approx^? Z_1 + 2Z_3 + Z_4 \\
Y_1 &\approx^? Z_3 + Z_4 + Z_5 + Z_7 \\
Y_2 &\approx^? Z_1 + Z_2 + 2Z_6
\end{aligned}
$$

Stickel-Fages AC-unification - all the possible cases

Decide whether we will include (set to 1) or not (set to 0) every "new" variable. Every "old" variable must be different than zero.

In our example, we have 2^7 possibilities of including/excluding the variables $Z_1, \ldots, Z_7$, but after observing that X_1, X_2, X_3, Y_1, Y_2 cannot be set to zero, only 69 cases remain.

Stickel-Fages AC-unification - dropping impossible cases

Drop the cases where the variables representing constants or subterms headed by a different AC function symbol are assigned to more than one of the "new" variables.

For instance, the potential new unification problem

$$\{X_1 \approx^? Z_6, X_2 \approx^? Z_4, X_3 \approx^? f(Z_1, Z_4),$$
$$Y_1 \approx^? Z_4, Y_2 \approx^? f(Z_1, Z_6, Z_6)\}$$

should be discarded as the variable X_3, which represents the constant a, cannot unify with $f(Z_1, Z_4)$.

Stickel-Fages AC-unification - dropping more cases

Replace "old" variables by the original terms they substituted and proceed with the unification.

Some new unification problems may be unsolvable and **will be discarded later**. For instance:

$$\{X \approx^? Z_6, Y \approx^? Z_4, a \approx^? Z_4, b \approx^? Z_4, \ Z \approx^? f(Z_6, Z_6)\}$$

Stickel-Fages AC-unification - solutions

In our example,

$$f(X, X, Y, a, b, c) \approx^? f(b, b, b, c, Z)$$

the solutions are:

$$\left\{ \begin{array}{l} \sigma_1 = \{Y \to f(b, b), Z \to f(a, X, X)\} \\ \sigma_2 = \{Y \to f(Z_2, b, b), Z \to f(a, Z_2, X, X)\} \\ \sigma_3 = \{X \to b, Z \to f(a, Y)\} \\ \sigma_4 = \{X \to f(Z_6, b), Z \to f(a, Y, Z_6, Z_6)\} \end{array} \right\}$$

Adapting first-order AC-unification to nominal AC-unification

We found a loop while solving nominal AC-unification problems using Stickel-Fages' Diophantine-based algorithm.

For instance

$$f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$$

Variables are associated as below:

U_1 is associated with argument X,
U_2 is associated with argument W,
V_1 is associated with argument $\pi \cdot X$, and
V_2 is associated with argument $\pi \cdot Y$.

Table of Solutions

The Diophantine equation associated is $U_1 + U_2 = V_1 + V_2$.

The table with the solutions of the Diophantine equations is shown below. The name of the new variables was chosen to make clearer the loop we will fall into.

Table 3: Solutions for the Equation $U_1 + U_2 = V_1 + V_2$

U_1	U_2	V_1	V_2	$U_1 + U_2$	$V_1 + V_2$	New variables
0	1	0	1	1	1	Z_1
0	1	1	0	1	1	W_1
1	0	0	1	1	1	Y_1
1	0	1	0	1	1	X_1

After `solveAC`

$$\{X \approx^? X_1, W \approx^? Z_1, \pi \cdot X \approx^? X_1, \pi \cdot Y \approx^? Z_1\}$$
$$\{X \approx^? Y_1, W \approx^? W_1, \pi \cdot X \approx^? W_1, \pi \cdot Y \approx^? Y_1\}$$
$$\{X \approx^? Y_1 + X_1, W \approx^? W_1, \pi \cdot X \approx^? W_1 + X_1, \pi \cdot Y \approx^? Y_1\}$$
$$\{X \approx^? Y_1 + X_1, W \approx^? Z_1, \pi \cdot X \approx^? X_1, \pi \cdot Y \approx^? Z_1 + Y_1\}$$
$$\{X \approx^? X_1, W \approx^? Z_1 + W_1, \pi \cdot X \approx^? W_1 + X_1, \pi \cdot Y \approx^? Z_1\}$$
$$\{X \approx^? Y_1, W \approx^? Z_1 + W_1, \pi \cdot X \approx^? W_1, \pi \cdot Y \approx^? Z_1 + Y_1\}$$
$$\{X \approx^? Y_1 + X_1, W \approx^? Z_1 + W_1, \pi \cdot X \approx^? W_1 + X_1, \pi \cdot Y \approx^? Z_1 + Y_1\}$$

After solving the linear Diophantine system

Seven branches are generated:

$B1 - \{\pi \cdot X \approx^? X\}, \sigma = \{W \mapsto \pi \cdot Y\}$

$B2 - \sigma = \{W \mapsto \pi^2 \cdot Y, X \mapsto \pi \cdot Y\}$

$B3 - \{f(\pi^2 \cdot Y, \pi \cdot X_1) \approx^? f(W, X_1)\}, \sigma = \{X \mapsto f(\pi \cdot Y, X_1)\}$

$B4 - \mathit{No\ solution}$

$B5 - \mathit{No\ solution}$

$B6 - \sigma = \{W \mapsto f(Z_1, \pi \cdot X), Y \mapsto f(\pi^{-1} \cdot Z_1, \pi^{-1} \cdot X)\}$

$B7 - \{f(\pi \cdot Y_1, \pi \cdot X_1) \approx^? f(W_1, X_1)\},$

$\quad \sigma = \{X \mapsto f(Y_1, X_1),\ W \mapsto f(Z_1, W_1), Y \mapsto f(\pi^{-1} \cdot Z_1, \pi^{-1} \cdot Y_1)\}$

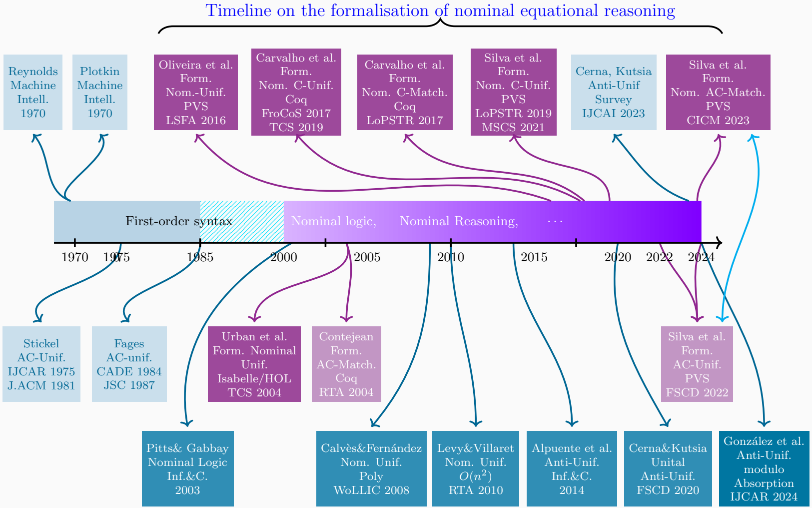
Focusing on Branch 7, notice that the problem before the AC Step and the problem after the AC Step and instantiating the variables are, respectively:

$$P = \{f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)\}$$

$$P_1 = \{f(X_1, W_1) \approx^? f(\pi \cdot X_1, \pi \cdot Y_1)\}$$

Work in Progress and Future Work

Synthesis on Nominal Equational Modulo



Results

		Synthesis Unification Nominal Modulo			
Theory	Unif. type	Equality-checking	Matching	Unification	Related work
$\approx_\alpha$	1	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	UPG04 LV10 CF08 CF10 LSFA2015
C	∞	$O(n^2 \log n)$	NP-comp.	NP-comp.	LOPSTR2017 FroCoS2017 TCS2019 LOPSTR2019 MSCS2021
A	∞	$O(n \log n)$	NP-comp.	NP-hard	LSFA2016 TCS2019
AC	ω	$O(n^3 \log n)$	NP-comp.	NP-comp.	LSFA2016 TCS2019 CICM2023

- Study how to avoid the circularity in nominal AC-unification.
  - How circularity enriches the set of computed solutions?
  - Under which conditions can circularity be avoided?
- Formalising anti-unification.
  - Only recently, anti-unification modulo α-, C-, and (αC)-symbols have been addressed. Procedures combining such properties have been shown to be challenging from theoretical and practical perspectives.

Thank you for your attention!

Thank you for your attention!

- Mauricio Ayala-Rincón, David M. Cerna, Andrés Felipe Gonzélez Barragán, and Temur Kutsia, *Equational Anti-unification over Absorption Theories*, IJCAR, 2024.
- María Alpuente, Santiago Escobar, Javier Espert, and José Meseguer, *A modular order-sorted equational generalization algorithm*, Information and Computation **235** (2014), 98–136.
- Mauricio Ayala-Rincón, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes Sobrinho, *A Certified Algorithm for AC-Unification*, FSCD, 2022.
- Franz Baader, *Unification, weak unification, upper bound, lower bound, and generalization problems*, RTA, 1991.

- Alexander Baumgartner, Temur Kutsia, Jordi Levy, and Mateu Villaret, *Higher-order pattern anti-unification in linear time*, J. Autom. Reason. **58** (2017), no. 2, 293–310.
- David M. Cerna, *Anti-unification and the theory of semirings*, Theo. Com. Sci. **848** (2020), 133–139.
- David M. Cerna and Temur Kutsia, *A generic framework for higher-order generalizations*, FSCD, 2019.
- ______, *Higher-order pattern generalization modulo equational theories*, Math. Struct. Comput. Sci. **30** (2020), no. 6, 627–663.
- ______, *Idempotent anti-unification*, ACM Trans. Comput. Log. **21** (2020), no. 2, 10:1–10:32.
- ______, *Unital anti-unification: type algorithms*, 2020.

- _______, *Anti-unification and generalization: A survey*, IJCAI, 2023.
- Gordon D. Plotkin, *A note on inductive generalization*, Machine Intelligence 5 **5** (1970), 153–163.
- John C. Reynolds, *Transformational system and the algebric structure of atomic formulas*, Machine Intelligence 5 **5** (1970), 135–151.