Nominal Equational Reasoning

Mauricio Ayala-Rincón[†],[‡]

Joint work with: Maribel Fernández[*], Gabriel Ferreira Silva[†], and Daniele Nantes Sobrinho[‡]

Salvador/Brasília/On-line, August 26th, 2020

http://nominal.cic.unb.br

Universidade de Brasília

Departments of [†]Computer Science and [‡]Mathematics

King's College London

[*]Department of Informatics

Table of contents

Motivation

Equational Problems

- **Equality check**: $s = t$?
- **Matching**: There exists σ such that $s\sigma = t$?
- **Unification**: There exists σ such that $s\sigma = t\sigma$?

s and t are *terms* in some *signature* and σ is a *substitution*.

Equational Problems - 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 $f(b, y)$

Equational Problems - 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 $\{x/b, y/a\}$.

Equational Problems - Syntactic unification

- $\mathcal{F}$ set of function symbols.
- $\mathcal{V}$ set of variables.
- x, y, z variables.
- a, b, c constant symbols.
- f, g, h function symbols.
- $\mathcal{T}(\mathcal{F}, \mathcal{V})$ set of terms over $\mathcal{F}$ and $\mathcal{V}$.
- s, t, u terms.
- $\sigma, \gamma, \delta : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ set of substitutions.

Substitutions have finite domain: $\{v \mid v\sigma \neq v\}$ is finite.

Equational Problems - Syntactic Unification

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:

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

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

Equational Problems - Syntactic Unification

Goal: *algorithm* that *unifies* terms.

Example:

- $h(\underbrace{x}, y, z) = h(\underbrace{f(w, w)}, f(x, x), f(y, y))$

Equational Problems - Syntactic Unification

Goal: *algorithm* that *unifies* terms.

Example:

- $h(\underbrace{x}, y, z) = h(\underbrace{f(w, w)}, f(x, x), f(y, y))$
- $h(f(w, w), \underbrace{y}, z) =$
  $h(f(w, w), \underbrace{f(f(w, w), f(w, w))}, f(y, y))$, partial solution:
  $\{x/f(w, w)\}$

Equational Problems - Syntactic Unification

Goal: *algorithm* that *unifies* terms.

Example:

- $h(\underbrace{x}, y, z) = h(\underbrace{f(w,w)}, f(x,x), f(y,y))$
- $h(f(w,w), \underbrace{y}, z) =$
  $h(f(w,w), \underbrace{f(f(w,w), f(w,w))}, f(y,y))$, partial solution:
  $\{x/f(w,w)\}$
- $h(f(w,w), f(f(w,w), f(w,w)), \underbrace{z}) =$
  $h(f(w,w), f(f(w,w), f(w,w)), \underbrace{f(f(f(w,w), f(w,w)), f(f(w,w), f(w,w)))})$,
  partial solution: $\{x/f(w,w), y/f(f(w,w), f(w,w))\}$

Equational Problems - Syntactic Unification

Goal: *algorithm* that *unifies* terms.

Example:

- $h(\underbrace{x}, y, z) = h(\underbrace{f(w,w)}, f(x,x), f(y,y))$
- $h(f(w,w), \underbrace{y}, z) =$
  $h(f(w,w), \underbrace{f(f(w,w), f(w,w))}, f(y,y))$, partial solution:
  $\{x/f(w,w)\}$
- $h(f(w,w), f(f(w,w), f(w,w)), \underbrace{z}) =$
  $h(f(w,w), f(f(w,w), f(w,w)), \underbrace{f(f(f(w,w), f(w,w)), f(f(w,w), f(w,w)))})$,
  partial solution: $\{x/f(w,w), y/f(f(w,w), f(w,w))\}$
- $h(f(w,w), f(f(w,w), f(w,w)), f(f(f(w,w), f(w,w)), f(f(w,w), f(w,w)))) =$
  $h(f(w,w), f(f(w,w), f(w,w)), f(f(f(w,w), f(w,w)), f(f(w,w), f(w,w))))$,
  solution: $\{x/f(w,w), y/f(f(w,w), f(w,w)), z/f(f(f(w,w), f(w,w)), f(f(w,w), f(w,w)))\}$.

Equational Problems - Syntactic Unification

Interesting questions:

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

Syntactic unification is of type *unary* and linear.

Equational Problems - 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*.

Equational Problems - 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*.

Equational Problems - Unification Modulo

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*.

Equational Problems - 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*.

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*.

Equational Problems - 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).

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

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

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

Equational Problems - 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).

Equational Problems - 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 i

		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

Nominal Syntax

Nominal syntax extends first-order syntax by bringing mechanisms to deal with bound and free variables in a natural manner.

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

Purpose of Presentation

- We revisit the contributions on nominal equational reasoning modulo associative and commutative operators and related work.
- We briefly comment about our work in progress on nominal AC-unification and its formalisation in PVS.

Nominal Syntax

Nominal Syntax

Nominal Terms, Permutations and Substitutions

Atoms and Variables

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

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:

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

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

$$s, t \quad ::= \quad \langle\rangle \;\mid\; a \;\mid\; \pi \cdot X \;\mid\; [a]t \;\mid\; \langle s, t\rangle \;\mid\; f\ t \;\mid\; f^E t$$

The symbols denote respectively: unit, atom term, suspended variable, abstraction, pair, function application and E-function application (E may be for A, C, AC, etc).

Examples of Permutation Actions

Permutations act on atoms and terms:

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

Nominal Syntax

Freshness and α-Equality

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

Freshness conditions $a\#s$, and *atom permutations* $\pi \cdot s$.

Example

β and η rules as nominal rewriting rules:

$$\boxed{\begin{array}{lr} app\langle lam[a]M, N\rangle \to subs\langle [a]M, N\rangle & (\beta) \\[1em] a\#M \vdash lam[a]app\langle M, a\rangle \to M & (\eta) \end{array}}$$

Some substitution rules:

$$\boxed{\begin{array}{l} b\#M \vdash subs\langle [b]M, N\rangle \to M \\[1em] a\#N \vdash subs\langle [b]lam[a]M, N\rangle \to lam[a]sub\langle [b]M, N\rangle \\[1em] c\#M, c\#N \vdash subs\langle [b]lam[a]M, N\rangle \to lam[c]sub\langle [b](a\ c) \cdot M, N\rangle \end{array}}$$

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 α-Equivalence

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

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

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

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

Additional Rule for α-Equivalence with C Functions

Let f be a C function symbol.

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

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

Additional Rule for α-Equivalence with AC Functions

Let f be an AC function symbol.

We add rule ($\approx_\alpha$ *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 $f*$.

The Operators S_n and D_n

Let f be an AC function:

- $S_2(f\langle f\langle a, b\rangle, f\langle [a]X, \pi \cdot Y\rangle\rangle)$ is equal to b.
- $D_2(f\langle f\langle a, b\rangle, f\langle [a]X, \pi \cdot Y\rangle\rangle\rangle)$ is equal to $f\langle f\ a, f\langle [a]X, \pi \cdot Y\rangle\rangle\rangle)$.

Derivation Rules as a Sequent Calculus

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 E-Unification and equational reasoning

Nominal E-Unification and equational reasoning

Nominal C-unification

Nominal C-unification

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

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.

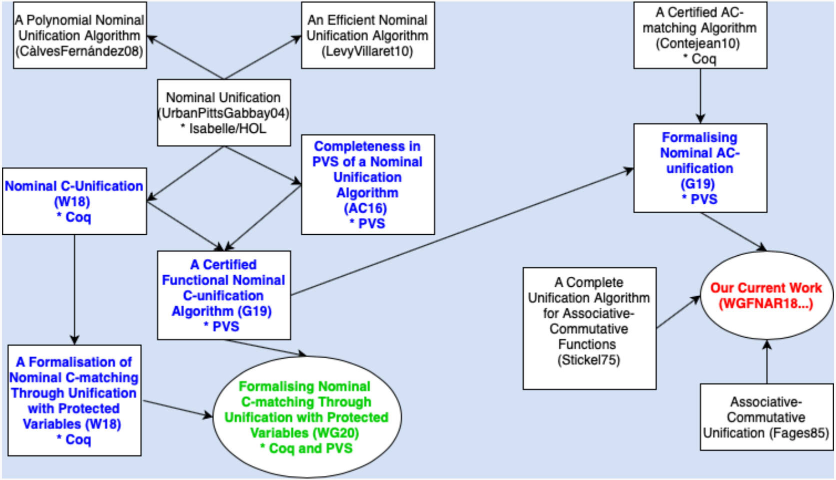
For instance, in a signature in which $\oplus$ and $\star$ are C, the unification problem: $\langle \emptyset, \{(a\ b)X \approx_\alpha{}^? X\}\rangle$

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

Nominal E-Unification and equational reasoning

Contextualisation of results

Contextualisation of results



Synthesis of results on Nominal Unification Modulo

		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 sub2020
A	∞	$O(n \log n)$	NP-comp.	NP-hard	LSFA2016 TCS2019
AC	ω	$O(n^3 \log n)$	NP-comp.	NP-comp.	LSFA2016 TCS2019

More on Nominal Reasoning

Also:

- Overlaps in Nominal Rewriting [LSFA 2015]
- Nominal Narrowing [FSCD 2016]
- Nominal Intersection Types [TCS 2018]
- Nominal Disequations [LSFA 2019]
- Nominal Syntax with Permutation Fixed Points [LMCS2020]

Co-authors: Washington R. de Carvalho, Ana Cristina Rocha Oliveira, Deivid Vale, Daniel Lima Ventura, Murdoch Gabbay.

Nominal reasoning with permutation fixed-point constraints

Instead of using freshness constraints ($a\#s$, for all $a \in dom(\pi)$), one uses *permutation fixed-point constraints*:

$$\pi \curlywedge s \text{ means that "}\pi \textit{ fixes "}s\text{"}$$

Solutions of the unification problem

$$(a\ b) \cdot X \oplus a \approx_\alpha{}^? Y \oplus X$$

using freshness constraints are:

$$\langle \emptyset, \{X/a, Y/b\}, \emptyset \rangle \text{ and } \langle \emptyset, \{Y/a\}, \{(a\ b) \cdot X \approx_\alpha X\} \rangle,$$

while using fixed-point constraints are:

$$\langle \emptyset, \{X/a, Y/b\} \rangle \text{ and } \langle \emptyset, \{Y/a, (a\ b) \curlywedge X\} \rangle.$$

Fixed-point constraints avoid infinite solutions as those related with fixed-point equations in the standard nominal approach.

Nominal E-Unification and equational reasoning

Progress on AC-unification

Example of Unification Problem and Solution

f is an AC function symbol.

One possible solution for

$$\langle \emptyset,\ f\langle f\langle X, Y\rangle, c\rangle\rangle \ \approx_? \ f\langle c, f\langle a, b\rangle\rangle\rangle$$

is:

$$\langle \emptyset, \{X \to a, Y \to b\}\rangle$$

What Is Tricky About AC? An Example

Let f be an AC function symbol.

- The solutions that come to mind when unifying

$$f(x, y) \approx^? f(a, z)$$

  are: $\{x/a, y/z\}$ and $\{x/z, y/a\}$.

What Is Tricky About AC? An Example

Let f be an AC function symbol.

- The solutions that come to mind when unifying

  $$f(x, y) \approx^? f(a, z)$$

  are: $\{x/a, y/z\}$ and $\{x/z, y/a\}$.

  Are there other solutions?

What Is Tricky About AC? An Example

Let f be an AC function symbol.

- The solutions that come to mind when unifying

$$f(x, y) \approx^? f(a, z)$$

  are: $\{x/a, y/z\}$ and $\{x/z, y/a\}$.

  Are there other solutions?

  Yes!

What Is Tricky About AC? An Example

Let f be an AC function symbol.

- The solutions that come to mind when unifying

$$f(x, y) \approx^? f(a, z)$$

  are: $\{x/a, y/z\}$ and $\{x/z, y/a\}$.

  Are there other solutions?

  Yes!

- For instance,
  $\{x/f(a, z_1), y/f(z_2), z/f(z_1, z_2)\}$ and
  $\{x/f(z_1), y/f(a, z_2), z/f(z_1, z_2)\}$.

What is Tricky About AC? New Variables and Termination

In the last example, new variables z_1 and z_2 were introduced.

Termination of syntactic unification relies on the decrease of the number of variables in the problem (while the domain of the substitution being built increases).

In a given step of AC-unification, the number of new variables introduced can be greater than the number of eliminated variables (did not happen in our example, but there are cases in which this happens).

The proof of termination is harder in AC-unification.

What is Tricky About AC? The Combinatory

If $s \equiv f^{AC}(s_1, ..., s_m)$ and $t \equiv f^{AC}(t_1, ..., t_n)$ are in flattened form:

- **Equality-Checking**: if $s = t$ then $m = n$ and for every s_i, there should be a corresponding t_j, such that $s_i = t_j$.
- **Matching**: if $s\sigma = t$, this **does not** mean that $s_i\sigma$ should correspond to some t_j.
- **Unification**: if $s\sigma = t\sigma$, this **does not** mean that $s_i\sigma$ should correspond to some $t_j\sigma$.

AC-Unification and Solving Linear Equations in $\mathbb{N}$

[Stickel 1975], [Stickel 1981], and [Fages 1987] propose an algorithm that uses a correspondence between unifying AC-functions and solving linear equations in $\mathbb{N}$.

Example (Stickel):

- Unification problem: $\boxed{f(x,x,y,a,b,c) \approx^? f(b,b,b,c,z)} \rightsquigarrow$ $f(x,x,y,a) \approx^? f(b,b,z) \rightsquigarrow f(x_1,x_1,x_2,x_3) \approx^? f(y_1,y_1,y_2)$
- Equation: $2x_1 + x_2 + x_3 = 2y_1 + y_2 \rightsquigarrow$ linear Diophantine system with new variables $\rightsquigarrow$

$$\text{Solutions: } \left\{ \begin{array}{l} \{y/f(b,b), z/f(a,x,x)\}, \\ \{y/f(z_2,b,b), z/f(a,z_2,x,x)\}, \\ \{x/b, z/f(a,y)\}, \\ \{x/f(z_6,b), z/f(a,y,z_6,z_6)\} \end{array} \right\}$$

Our Current Work

- Stickel and Fages' approach computes solutions in such a manner that the generation of redundancies is avoided building a *minimal complete set* of AC-unifiers.
- In UNIF 2019 we presented a functional specification for nominal AC-unification that was formalised sound in PVS. The algorithm follows a *combinatorial* approach that was redundant and unable to provide a minimal complete set of AC-unifiers. Currently, we are following Stickel and Fages' method to guarantee also *minimality* and *completeness*.

http://nominal.cic.unb.br

Conclusion and Future Work

Conclusion - Formalisations

- Functional nominal α-unification was formalised (PVS).
- Nominal A-, C-, and AC- equality-check were formalised and implemented (Coq, OCaml).
- Nominal C-matching and C-unification were formalised and implemented (Coq, PVS, OCaml, Python, LISP)
- Nominal AC-matching and AC-unification formalisation is under development (PVS).

Future Work

Future work:

- There is a lot to be done to develop and formalise Nominal Equational Reasoning: not only to deal with unification modulo other theories as ACh and ACUh, AI, AUCUN, etc, but also to establish nominal anti-unification, disunification, symmetric unification, etc.
- Developing nominal rewriting and nominal type systems.
- Such nominal mechanisms are indeed relevant in practice in frameworks such as the PVS and Isabelle/HOL nominal developments, and computational programming and deductive tools such as αProlog, α-Check, Cαml, etc.

Thank You

Thank you! Any questions?

Appendix - Example of Stickel's Algorithm for AC-Unification

Example

How do we generate all solutions to solve:

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

1. Eliminate common arguments
   $f(X, X, Y, a, \cancel{b}, \cancel{c}) \approx^? f(\cancel{b}, b, b, \cancel{c}, Z)$

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

2. Generalise substituting distinct arguments by new variables:

   $$\rightsquigarrow \; f(X_1, X_1, X_2, X_3) \approx^? f(Y_1, Y_1, Y_2)$$

   .

Example - Auxiliar Algorithm

3. Apply an auxiliar algorithm that unifies AC-function symbols with only variables as arguments.

After this big step (detailed in the next slides) we will have 69 cases.

Example - Introducing Diophantine Equations

3.1. Transform the unification problem into a linear Diophantine equation.

After this step, our equation is

$$\leadsto \quad 2X_1 + X_2 + X_3 = 2Y_1 + Y_2$$

Example - Basis of Solutions

3.2. Generate a basis of solutions to the system of Diophantine equations.

After this step we have Table 3:

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

Example - Associating New Variables

3.3. Associate new variables with each solution.

After this step we have:

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

Example - Old and New Variables

3.4. Relate the "old" and the "new" variables.

After this step, we obtain:

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

Example - All the Possible Cases

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

In our example, we have $2^7 = 128$ possibilities of including/excluding the new variables $Z_1, \ldots, Z_7$, but after observing that the old variables $X1, X_2, X_3, Y_1, Y_2$ cannot be set to 0, only 69 cases remain.

Example - Dropping Impossible Cases

4. The fourth step is to drop the cases where the variables that in fact represent constants and subterms headed by a different AC function symbol are assigned to more than one of the "new" variables.

For instance, the potential solution

$$\sigma_{wrong} = \{X_1 \to Z_6, X_2 \to Z_4, X_3 \to f(Z_1, Z_4), \\ Y_1 \to Z_4, Y_2 \to f(Z_1, Z_6, Z_6)\}$$

should be discarded as the variable X_3, which represents the constant a, must not be assigned to $f(Z_1, Z_4)$.

Example - Dropping More Cases

5. Replace variables by the original terms they substituted. Drop cases where a "new" variable is being mapped to two or more "old" variables that in fact represent different constants.

In our example, the potential solution

$$\sigma = \{X_1 \to Z_6, X_2 \to Z_4, X_3 \to Z_4, Y_1 \to Z_4, Y_2 \to f(Z_6, Z_6)\}$$

shoud be discarded, as the variables X_3 and Y_1, representing the constants a and b, cannot be mapped to the same "new" variable Z_4.

Example - Normalising Solutions

6. Normalise remaining cases by replacing the variables in the image of the substitution that also happen in the domain. Output the solutions.

In our example, the solutions are:

$$\left\{ \begin{array}{c} \{Y \to f(b, b), Z \to f(a, X, X)\} \\ \{Y \to f(Z_2, b, b), Z \to f(a, Z_2, X, X)\} \\ \{X \to b, Z \to f(a, Y)\} \\ \{X \to f(Z_6, b), Z \to f(a, Y, Z_6, Z_6)\} \end{array} \right\}$$