Nominal AC-Matching

KCL - UIUC 6th Meeting

Gabriel Ferreira Silva (Universidade de Brasília - UnB)
Advisor: Mauricio Ayala-Rincón (Universidade de Brasília - UnB)
Co-Advisor: Maribel Fernández (King's College London)

https://gabriel951.github.io/

Joint Work With

Figure: Mauricio Ayala-Rincón

Figure: Maribel Fernández

Figure: Daniele Nantes

Figure: Temur Kutsia

Outline

Unification

Unification is about "finding a way" to make two terms equal:

- $f(a, X)$ and $f(Y, b)$ can be made equal by "sending" X to b and Y to a, as they both become $f(a, b)$.

Unification has a lot of applications: logic programming, theorem proving, type inference and so on.

Unification Modulo AC

AC-unification is unification in the presence of associative-commutative function symbols.

For instance, if f is an AC function symbol, then:

$$f(a, f(b, c)) \approx f(c, f(a, b)).$$

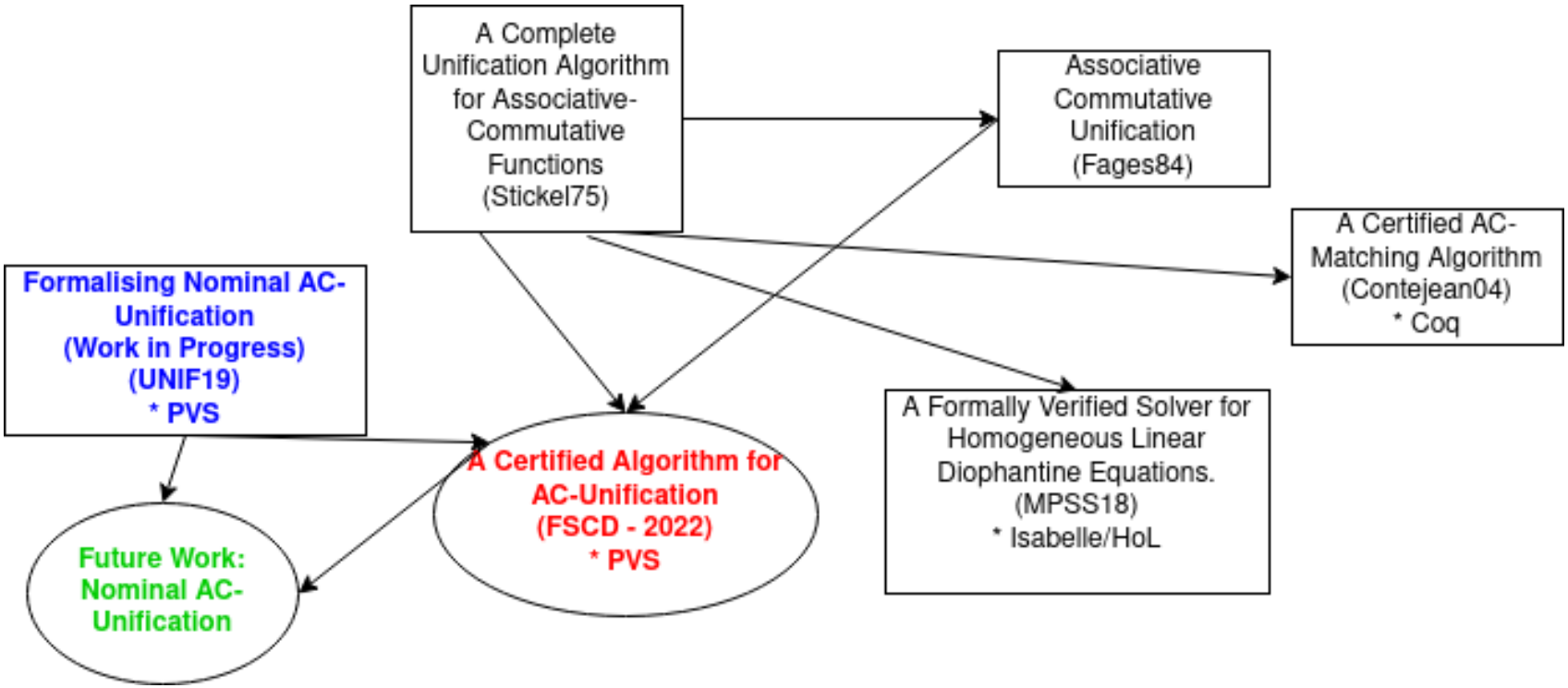
In this Talk

1. Briefly discuss first-order AC-unification.
2. Explain the nominal setting and describe the obstacles towards a nominal AC-unification algorithm.
3. Discuss our plan to formalise nominal AC-matching.

Our Work in First Order AC-Unification in a Nutshell

We modified Stickel's seminal AC-unification algorithm to avoid mutual recursion and formalised it in the PVS proof assistant. We proved the adjusted algorithm's termination, soundness and completeness [AFSS22].

Main Related Work



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: $\{X \to a, Y \to Z\}$ and $\{X \to Z, Y \to a\}$.

Are there other solutions?

What is Tricky About AC? An Example

Yes!

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

The AC Step for AC-Unification

We explain via an example the **AC Step** for AC-unification.

How do we generate a complete set of unifiers for:

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

Eliminate Common Arguments

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

Now we must unify $f(X, X, Y, a)$ with $f(b, b, Z)$.

Introducing a Linear Equation on $\mathbb{N}$

According to the number of times each argument appear in the terms, transform the unification problem into a linear equation on $\mathbb{N}$.

After this step, our equation is:

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

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

Basis of Solutions

Generate a basis of solutions to the linear equation.

Table: 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

Associating New Variables

Associate new variables with each solution.

Table: 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

Old and New Variables

Observing Table 2, relate the "old" variables and the "new" ones.

After this step, we obtain:

$$X_1 \approx^? Z_6 + Z_7$$
$$X_2 \approx^? Z_2 + Z_4 + 2Z_5$$
$$X_3 \approx^? Z_1 + 2Z_3 + Z_4$$
$$Y_1 \approx^? Z_3 + Z_4 + Z_5 + Z_7$$
$$Y_2 \approx^? Z_1 + Z_2 + 2Z_6$$

All the Possible Cases

Decide whether we will include (set to 1) or not (set to 0) 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 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, we have 69 cases.

Dropping Impossible Cases

Drop the cases where the variables that in fact represent 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 $f(Z_1, Z_4)$.

Dropping More Cases and Proceeding

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:

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

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

In our example, the solutions will be:

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

Input and Output of Algorithm

The algorithm `ACUnif` is recursive and receives as input a triple (P, σ, V), where P is the current unification problem, σ is the substitution computed so far and V are the variables that are/were in the problem.

To unify two terms t and s the initial call is with $P = \{t \approx^? s\}$, $\sigma = Id$ and $V = Vars(t, s)$.

The output is a list of substitutions, where each substitution δ in this list is an AC-unifier of P.

Termination, Soundness and Completeness

We have proved termination, soundness and completeness:

- Termination - Hard
- Soundness - Easy
- Completeness - Hard

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 $\exists x : x \geq 0$ and $\exists y : y \geq 0$ are not syntactically equal, but should be considered equivalent in a system with binding.

Nominal

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

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 $\mathbb{A} = \{a, b, c, \ldots\}$.

Nominal Terms

Nominal terms are inductively generated according to the grammar:

$$t, s ::= a \mid \pi \cdot X \mid [a]t \mid f(t_1, \ldots, t_n)$$

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

Freshness predicate

$a\#t$ means that if a occurs in t then it does so under an abstractor $[a]$.

A context is a set of constraints of the form $a\#X$. Contexts are denoted as δ, ∇ or γ.

An Example on Nominal Rewriting I

We can define[1] a signature for first-order logic with term-formers $\forall, \exists, \neg, \wedge, \vee$ and rewrite rules to compute prenex normal forms (here we use variables P, Q):

$$a\#P \vdash P \wedge \forall[a]Q \to \forall[a](P \wedge Q)$$
$$a\#P \vdash (\forall[a]Q) \wedge P \to \forall[a](Q \wedge P)$$
$$a\#P \vdash P \vee \forall[a]Q \to \forall[a](P \vee Q)$$
$$a\#P \vdash (\forall[a]Q) \vee P \to \forall[a](Q \wedge P)$$

An Example on Nominal Rewriting II

$$
\begin{aligned}
a\#P &\vdash P \wedge \exists[a]Q \rightarrow \exists[a](P \wedge Q) \\
a\#P &\vdash (\exists[a]Q) \wedge P \rightarrow \exists[a](Q \wedge P) \\
a\#P &\vdash P \vee \exists[a]Q \rightarrow \exists[a](P \vee Q) \\
a\#P &\vdash (\exists[a]Q) \vee P \rightarrow \exists[a](Q \vee P) \\
&\vdash \neg(\exists[a]Q) \rightarrow \forall[a]\neg Q \\
&\vdash \neg(\forall[a]Q) \rightarrow \exists[a]\neg Q
\end{aligned}
$$

[1] Example taken from [FG07].

Adapting to Nominal

We believe it won't be too hard to adapt the proofs of soundness and completeness to nominal AC-unification.

Fixpoint Equations $\pi \cdot X \approx^? X$

Nominal Unification - $\pi \cdot X \approx^? X$ is solved by adding $dom(\pi)\#X$ to our context.

Nominal C-Unification - There are infinite solutions to $\pi \cdot X \approx^? X$, and there is an enumeration procedure to do it (see [ARCSFNS17]). In the algorithm for nominal C-unification, equations such as $\pi \cdot X \approx^? X$ are part of the output.

Nominal AC-Unification - Work in progress, similar to nominal C-unification (more details in Appendix).

Termination

Termination will be harder. Equations such as

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

give us a loop.

The Loop

After solving the corresponding Diophantine equation, we generate 7 branches. One of them is:

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

and after we instantiate the variables that we can we get:

$$P_1 = \{f(\pi \cdot Y_1, \pi \cdot X_1) \approx^? f(W_1, X_1)\},$$
$$\sigma = \{X \mapsto f(Y_1, X_1),\ W \mapsto f(Z_1, W_1), Y \mapsto f(\pi^{-1} \cdot Z_1, \pi^{-1} \cdot Y_1)\}$$

The Loop

The problem before and after are respectively:

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

Characterizing Problematic Equations

If $f(t_1, \ldots, t_m) \approx^? f(s_1, \ldots, s_n)$ and $\pi_1 \cdot X \in \{t_1, \ldots, t_m\}$ and $\pi_2 \cdot X \in \{s_1, \ldots, s_n\}$, we may get a loop.

Solutions in the nominal setting

Nominal Unification - Algorithm output is a pair $\langle \nabla, \sigma \rangle$.

Nominal C-Unification - Fixed-point equations (e.g. $\pi \cdot X \approx^? X$) are included in the algorithm output, which is a set of triples $S = \{\langle \nabla_1, \sigma_1, FP_1 \rangle, \ldots, \langle \nabla_n, \sigma_n, FP_n \rangle\}$.

Nominal AC-Unification - Should we include equations like $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$ as part of the algorithm output? (more details in Appendix).

Focus on Nominal AC-Matching

Due to time constraints (my PhD finishes in the middle of 2023), we thought of switching the focus from nominal AC-unification to **nominal AC-matching**.

Advantages:

- Important problem, with applications such as nominal rewriting.
- Should be easier than nominal AC-unification.

Matching and Unification

Matching can be seen as an easier version of unification, where the terms in the right-hand side do not contain variables.

From unification to matching via $\mathcal{X}$

Given an algorithm of unification, one can adapt it by adding as a parameter a set of protected variables $\mathcal{X}$, which cannot be instantiated.

The adapted algorithm can then be used for:

- **Unification** - By putting $\mathcal{X} = \emptyset$
- **Matching** - By putting $\mathcal{X}$ as the set of variables in the right-hand side
- α-**Equivalence** - By putting $\mathcal{X}$ as the set of variables that appear in the problem.

OBS: This approach was taken when adapting a nominal C-unification algorithm to handle matching (see [AdCSF+21]).

Another option for nominal AC-Matching

Another option: formalise a nominal AC-matching algorithm from scratch.

Approaches for nominal AC-matching

There are two possible approaches for formalising nominal AC-matching:

- Nominal AC-Matching via unification and a protected set of variables $\mathcal{X}$.
- From scratch.

We feel like the first approach could be used on future works to reason about nominal AC-unification. Since we already have a first-order AC-unification algorithm formalised it may also be easier than starting from scratch.

Plan for Nominal AC-Matching

A two step plan for nominal AC-matching:

- Step 1: Extend the formalisation of first-order AC-unification to also handle first-order AC-matching, via protected variables.
- Step 2: Extend the formalisation obtained in Step 1 to nominal.

Nominal AC-Matching

Things to Worry About:

- Do we avoid problems like $f(X, W) \approx^{?} f(\pi \cdot X, \pi \cdot Y)$?
- Termination
- Soundness and Completeness

Do we avoid problems like $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$? I

Initially, in our matching problem, all the variables in the right-hand side are protected.

But when we start introducing the new variables Z_is, can we get a problem as $f(t_1, \ldots, t_m) \approx^? f(s_1, \ldots, s_n)$, where $\pi_1 \cdot X \in \{t_1, \ldots, t_m\}$ and $\pi_2 \cdot X \in \{s_1, \ldots, s_n\}$ and $X \notin \mathcal{X}$?

Do we avoid problems like $f(X, W) \approx^{?} f(\pi \cdot X, \pi \cdot Y)$? II

Idea: Prove that every new variable Z_i introduced in the AC Step will be instantiated.

Do we avoid problems like $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$? III

Suppose the problem is $f(s_1, \ldots, s_m) \approx^? f(t_1, \ldots, t_n)$, where t_i is not an unprotected variable, for $1 \leq i \leq n$.

Every new variable Z_i is associated with row i of the Diophantine matrix D, which is a non-zero solution to the Diophantine equation. Hence at least one column $j + m$ that corresponds to a term t_j on the right-hand side will have it's i-th entry non-zero, i.e.:

$$D[i, j + m] \neq 0$$

Do we avoid problems like $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$? IV

Hence, one of the unification problems generated will be: $t_j \approx^? Z_i$ and the new variable Z_i will be instantiated.

Do we avoid problems like $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$? V

Why can't we actually have $t_j \approx^? f(Z_i, \text{something else})$?

Because t_j is not an unprotected variable and is not an AC-function headed by f so this kind of equation has no solution and is eliminated by our algorithm.

Termination

Given a unification problem P, I am considering using as termination the lexicographic measure:

$$(Vars(P), size(P))$$

Soundness and Completeness of Nominal AC-Matching and AC-Unification

We expect the proofs of soundness and completeness of nominal AC-matching to be a straightforward adaptation from their first-order counterparts.

The proofs of soundness and completeness could be reused for nominal AC-unification.

Thank You

Thank you! Any comments/suggestions/doubts? [2]

[2] to see more of my work, visit https://gabriel951.github.io/.

References I


Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes-Sobrinho, *Formalising nominal C-unification generalised with protected variables*, Math. Struct. Comput. Sci. (2021).


Mauricio Ayala-Rincón, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes Sobrinho, *A Certified Algorithm for AC-Unification*, Formal Structures for Computation and Deduction, FSCD 2022 (2022).


M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho, *On Solving Nominal Fixpoint Equations*.


Maribel Fernández and Murdoch Gabbay, *Nominal rewriting*, Information and Computation (2007).

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

We found a loop while solving $f(X, W) \approx^? f(\pi \cdot X, \pi \cdot Y)$.

Table of Solutions

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.

The Diophantine equation associated[3] is $U_1 + U_2 = V_1 + V_2$ and the table of solutions is:

Table: 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

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

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 instantiating the variables

7 branches are generated:

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

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

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

$B4 - \textit{No solution}$

$B5 - \textit{No solution}$

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

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

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

The Loop

Focusing on *Branch7*, notice that the problem before `solveAC` and the problem after `solveAC` and instantiating the variables are:

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

Is $f(X, W) \approx^{?} f(\pi X, \pi Y)$ finitary?

Is there a finite set of triples $\langle \nabla, \sigma, FP \rangle$ that solve $f(X, W) \approx^{?} f(\pi X, \pi Y)$?

As will be shown in the next slides, the answer is yes.

Branch 3 is also a loop

Branch 3 also give us a loop and this can be seen more clearly if we write the result of taking branch 3 as:

$$P_1 = \{f(X_1, W_1) \approx^? f(\pi X_1, \pi Y_1)\},$$
$$\sigma_{B3} = \{X_0 \mapsto f(Y_1, X_1), W_0 \mapsto W_1, Y_0 \mapsto \pi^{-1} Y_1\}$$

OBS: We are going to consider $X = X_0, W = W_0$ and $Y = Y_0$.

Output of the algorithm and solutions

Let k be the order of π. I will show that it's enough for our algorithm to take branches 3 and 7 at most $2k$ times.

The output of the algorithm will be triples $\langle \emptyset, \sigma, FP \rangle$ such that σ is of the form $\sigma_{By}\sigma_{Bx_n} \ldots \sigma_{Bx_1}$, where x_i is either 3 or 7 and y is different than 3 or 7.

A solution $\langle \Delta, \delta \rangle$ to $f(X, W) \approx^? f(\pi X, \pi Y)$ is such that δ is of the form $\delta' \delta_{By}\delta_{Bx_m} \ldots \delta_{Bx_1}$, where x_i is either 3 or 7 and y is different than 3 or 7.

σ_{B3} and σ_{B7}

At the i-th iteration, the substitutions σ_{B3} and σ_{B7} differ only by the fact that σ_{B7} introduces variables Z_{i+1}:

$$\sigma_{B3} = \{X_i \mapsto f(Y_{i+1}, X_{i+1}), W_i \mapsto W_{i+1}, Y_i \mapsto \pi^{-1} Y_{i+1}\}$$
$$\sigma_{B7} = \{X_i \mapsto f(Y_{i+1}, X_{i+1}), W_i \mapsto f(Z_{i+1}, W_{i+1}),$$
$$Y_i \mapsto f(\pi^{-1} Z_{i+1}, \pi^{-1} Y_{i+1})\}$$

Notation

$[\pi_1, \ldots, \pi_n]X$ is syntactic sugar for $\pi_1 X, \ldots, \pi_n X$. Hence, the term denoted as $f([\pi_1, \ldots, \pi_n]Y, Z)$ is the term $f(\pi_1 Y, \ldots, \pi_n Y, Z)$.

Examples of the optional argument notation

If a term is of the form $f(Z^o, X, Y)$ then the term is either $f(X, Y)$ or $f(Z, X, Y)$.

If a term is of the form $f(Z^o, X^o, Y)$ then the term is one of:

- Y
- $f(Z, Y)$
- $f(X, Y)$
- $f(Z, X, Y)$

Finally, if a term is of the form $f([Id, \pi, \pi^2]Z^o, X, Y)$ then either ALL the arguments $Id\ Z, \pi Z, \pi^2 Z$ are in the term or NONE of them are. Hence, the term is either $f([Id, \pi, \pi^2]Z, X, Y)$ or $f(X, Y)$.

σ_{Bx_i}

The optional argument notation let us write σ_{Bx_i} as

$$\{X_i \mapsto f(Y_{i+1}, X_{i+1}), W_i \mapsto f(Z^o_{i+1}, W_{i+1}), Y_i \mapsto f(\pi^{-1} Z^o_{i+1}, \pi^{-1} Y_{i+1})\}$$

whether x_i is equal to 3 or 7.

$\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ and $\delta_{Bx_m} \ldots \delta_{Bx_1}$ I

Let's calculate $\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ applied to X_0, W_0 and Y_0.

$\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ and $\delta_{Bx_m} \ldots \delta_{Bx_1}$ II

$$
\begin{aligned}
X_0 &\mapsto f(Y_1, X_1) \\
&\mapsto f(\pi^{-1} Z_2^o, [\pi^{-1}, Id] Y_2, X_2) \\
&\mapsto f(\pi^{-1} Z_2^o, [\pi^{-2}, \pi^{-1}] Z_3^o, [\pi^{-2}, \pi^{-1}, Id] Y_3, X_3) \\
&\vdots \\
&\mapsto f(\pi^{-1} Z_2^o, \ldots, [\pi^{-(n-1)}, \ldots, \pi^{-1}] Z_n^o, [\pi^{-(n-1)}, \ldots, Id] Y_n, X_n)
\end{aligned}
$$

$\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ and $\delta_{Bx_m} \ldots \delta_{Bx_1}$ III

$$
\begin{aligned}
W_0 &\mapsto f(Z_1^o, W_1) \\
&\mapsto f(Z_1^o, Z_2^o, W_2) \\
&\vdots \\
&\mapsto f(Z_1^o, Z_2^o, \ldots, Z_n^o, W_n)
\end{aligned}
$$

$\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ and $\delta_{Bx_m} \ldots \delta_{Bx_1}$ IV

$$
\begin{aligned}
Y_0 &\mapsto f(\pi^{-1} Z_1^o, \pi^{-1} Y_1) \\
&\mapsto f(\pi^{-1} Z_1^o, \pi^{-2} Z_2^o, \pi^{-2} Y_2) \\
&\vdots \\
&\mapsto f(\pi^{-1} Z_1^o, \pi^{-2} Z_2^o, \ldots, \pi^{-n} Z_n^o, \pi^{-n} Y_n)
\end{aligned}
$$

The computation for $\delta_{Bx_m} \ldots \delta_{Bx_1}$ is analogous, replacing n by m.

Is there a substitution more general than $\delta_{Bx_m} \ldots \delta_{Bx_1}$?

Pick n such that $k \leq n < 2k$ and $n \equiv m \ (mod\ k)$. Consider the substitution $\sigma^* = \sigma_{Bx_n} \ldots \sigma_{Bx_1}$, where

$$\sigma_{Bx_i} = \begin{cases} \sigma_{B7}, & \text{if } i \leq k \text{ and} \\ & \exists j : j \equiv i \ (mod\ k) \text{ and } Z_j \in Args(\delta_{Bx_m} \ldots \delta_{Bx_1} W_0) \\ \sigma_{B3}, & \text{otherwise} \end{cases}$$

$\sigma^* \leq \delta_{Bx_m} \ldots \delta_{Bx_1}$ I

We can find λ such that $\delta_{Bx_m} \ldots \delta_{Bx_1} = \lambda\sigma^*$. Define λ by:

- If $i \leq k$ and $\exists j : j \equiv i \ (mod\ k)$ then $\lambda Z_i \mapsto f(Z_i, Z_{j_1}, \ldots, Z_{j_l})$, where $j_1, \ldots, j_l$ are all the indices that are equal to i modulo k such that $Z_{j_1}, \ldots, Z_{j_l}$ appear in $Args(\delta_{Bx_m} \ldots \delta_{Bx_1} W_0)$
- Otherwise, $\lambda Z_i \mapsto Z_i$

$\sigma^* \leq \delta_{Bx_m} \ldots \delta_{Bx_1} \mid\mid$

- $\lambda Y_n \mapsto Y_m$
- $\lambda W_n \mapsto W_m$

$\sigma^* \leq \delta_{Bx_m} \ldots \delta_{Bx_1}$ III

- Given a variable Z_j, let i_j be the index such that $i_j \leq k$ and $i_j \equiv j \ (mod \ k)$. Then,

$$\lambda X_n \mapsto f([\pi^{-i_{k+1}}, \ldots, \pi^{-k}]Z^o_{k+1}, \ldots, [\pi^{-i_n}, \ldots, \pi^{-(n-1)}]Z^o_n, \\ [\pi^{-n}, \ldots, \pi^{-(m-1)}]Y_m, X_m)$$

A set of triples is enough I

We only need to output the set of triples generated after taking branches 3 or 7 at most $2k$ times and then taking another branch.

A triple output by the algorithm in this case is of the form $\langle \emptyset, \sigma_{By}\sigma_{Bx_n} \ldots \sigma_{Bx_1}, FP_{By} \rangle$, where x_i is either 3 or 7 and y is different than 3 or 7 and $n \leq 2k$.

A set of triples is enough II

If δ is of the form $\delta' \delta_{By_i} \delta_{Bx_m} \ldots \delta_{Bx_1}$, then the triple output by the algorithm that we are looking for would be $(\emptyset, \sigma_{By_i} \sigma^*, FP_{By_i})$, where FP_{By_i} would be the fixpoint equation of branch y_i (it may be empty).

Lesson Learned

We don't need to include an equation like $f(X, W) \approx^? f(\pi X, \pi Y)$ in the output of our algorithm. A set of triples $\langle \nabla, \sigma, FP \rangle$ is enough!

Is this always the case? If we have $f(t_1, \ldots, t_m) \approx^? f(s_1, \ldots, s_n)$ and there exists $\pi_1 X \in Args(t)$ and $\pi_2 X \in Args(s)$, is a set of triples always enough? Can we generalise the argument we used for $f(X, W) \approx^? f(\pi X, \pi Y)$?

Difficulties in Generalising our Reasoning

- We may not get exactly a loop after applying `solveAC` and `instantiateStep`. For instance, adapting Stickel's example we may have :

$$P_0 = \{f(2X_1, X_2, X_3) \approx^? f(2\pi X_2, Y_1)\}$$
$$P_1 = \{f(\pi Z_2, \pi Z_4, 2\pi Z_5) \approx^? f(Z_3, Z_4, Z_5, Z_7)\}$$

- There may be more than one "doubly" suspended variable.

Generating all Solutions to $\pi X \approx^? X$

Can we generate all solutions to $\pi X \approx^? X$?

Changing our View

Solving $\pi X \approx^? X$ is equivalent to finding all the terms t such that there is a context Γ such that $\Gamma \vdash \pi t \approx^? t$.

A trivial procedure

Generate every term t and then find (if possible) the minimal context ∇ such that $\nabla \vdash \pi t \approx^{?} t$.

Enumerate all Solutions

Let's try to find a more interesting procedure. What should we aim for when solving fixpoint equations?

Two step plan:

1. An enumeration procedure SOLVEFIXPOINT that enumerates all solutions
2. From the enumeration procedure, put bounds in the number of recursive calls to obtain a terminating algorithm.

Enumeration Procedure I

The enumeration procedure will be given as a set of non-deterministic rules, that operate on triples of the form (Γ, σ, FP), where FP is a set of fixpoint equations we have to solve and of freshness problems we have to solve.

The initial call will be with the triple $(\emptyset, Id, \{\pi X \approx^? X\})$.

Enumeration Procedure II

Rules of the enumeration procedure:

- `(Var)`
- `(Func)`
- `(Abs a)` and `(Abs b)`
- `(AC Func)`
- Old rules for solving freshness problems
- `(Term)`

Enumeration Procedure III

The freshness problems are introduced by rule `(Abs b)`.

As we go applying the enumeration rules, no variable X appear in more than one fixpoint equation.

Variable Rule

`(Var)` rule:

$$(\Gamma, \sigma, \{\pi X \approx^? X\} \cup FP) \stackrel{Var}{\Longrightarrow} (\Gamma \cup dom(\pi)\#X, \sigma, FP)$$

Syntactic Function Rule

Let g be an arbitrary syntactic function symbol of arity m and let $\sigma' = \{X \mapsto g(X_1, \ldots, X_m)\}$, where $X_1, \ldots, X_m$ are **new** variables.

The `(Func)` rule:

$$(\Gamma, \sigma, \{\pi X \approx^? X\} \cup FP) \stackrel{Func}{\Longrightarrow}$$

$$(\Gamma, \sigma'\sigma, \{\pi X_1 \approx^? X_1, \ldots, \pi X_m \approx^? X_m\} \cup \sigma' FP)$$

Abstraction Rule - First case

Let $a \notin dom(\pi)$. Let $\sigma' = \{X \mapsto [a]X_1\}$, where X_1 is a **new** variable.
The `(Abs a)` rule:

$$(\Gamma, \sigma, \{\pi X \approx^? X\} \cup FP) \xRightarrow{Abs\ a} (\Gamma, \sigma'\sigma, \{\pi X_1 \approx^? X_1\} \cup \sigma' FP)$$

Abstraction Rule - Second case

Let $a \in dom(\pi)$. Let $\pi' = (a\ \pi a)\ \pi$ and let $\sigma' = \{X \mapsto [a]X_1\}$, where X_1 is a **new** variable.
The `(Abs b)` rule:

$$(\Gamma, \sigma, \{\pi X \approx^? X\} \cup FP) \xrightarrow{Abs\ b}$$

$$(\Gamma, \sigma'\sigma, \{\pi' X_1 \approx^? X_1\} \cup \sigma' FP \cup \{a\#_?\pi X_1\})$$

AC Function rule I

Let m be an arbitrary number and let ψ be an arbitrary permutation from $\{1, \ldots, m\}$ to $\{1, \ldots, m\}$, such that:

$$\psi = (x_1 x_2 \ldots x_{m_1})(x_{m_1+1} x_{m_1+2} \ldots x_{m_2}) \ldots (x_{m_{k-1}+1} x_{m_{k-1}+2} \ldots x_{m_k})$$

and let $l_1, \ldots, l_k$ be the length of the cycles.

Let σ' be:

$$\sigma' = X \mapsto f(\underbrace{X_1, \pi^1 X_1 \ldots, \pi^{l_1-1} X_1}, \ldots, \underbrace{X_k, \pi^1 X_k, \ldots, \pi^{l_k-1} X_k})$$

AC Function rule II

The `(AC Func)` rule is:

$$(\Gamma, \sigma, \{\pi X \approx^? X\} \cup FP) \xRightarrow{ACFunc}$$
$$(\Gamma, \sigma'\sigma, \{\pi^{l_1} X_1 \approx^? X_1, ..., \pi^{l_k} X_k \approx^? X_k\} \cup \sigma' FP)$$

Termination rule

`(Term)` rule:

$$(\Gamma, \sigma, \emptyset) \xRightarrow{\mathit{Term}} (\Gamma, \sigma)$$

Solution when t is an Atom

A solution is when $t = \sigma X$ is an atom $a_i \notin dom(\pi)$:

$$\langle \emptyset, X \mapsto a_i \rangle$$

Notice that this solution, however, is less general than $\langle dom(\pi) \# X, Id \rangle$ if we consider the substitution $\sigma' = X \mapsto a_i$. Therefore, there is no need for a rule for atoms.

Example 1 I

Let $*$ and $+$ be AC-function symbols and $\pi = (123456)$. Consider the solution:

$$\langle \emptyset, X \mapsto *(+(1,4), +(2,5), +(3,6)) \rangle$$

How can we inductively generate this solution?

Example 1 II

1. In the first iteration we may consider $m = 3$ and the permutation $\psi = (123)$. Then, we would instantiate $X \mapsto *(X_1, \pi^1 X_1, \pi^2 X_1)$ and proceed to solve $\pi^3 X_1 \approx^? X_1$.

2. In the second iteration we may consider $m = 2$ and the permutation $\psi = (12)$. Our algorithm would instantiate $X_1 \mapsto +(X_2, \pi^3 X_2)$ and proceed to solve $(\pi^3)^2 X_2 \approx^? X_2$.

Example 1 III

3. In the third iteration, notice that $(\pi^3)^2 = \pi^6 = Id$. The solution to $\pi^6 X_2 \approx^? X_2$ would be $\langle \emptyset, Id \rangle$.

4. Plugging this value back, we would generate the solution

$$\langle \emptyset, X \mapsto *(+(X_2, \pi^3 X_2), +(\pi X_2, \pi^4 X_2), +(\pi^2 X_2, \pi^5 X_2)) \rangle$$

Example 1 IV

The particular solution:

$$\langle \emptyset, X \mapsto *(+(1,4), +(2,5), +(3,6)) \rangle$$

can be obtained from:

$$\langle \emptyset, X \mapsto *(+(X_2, \pi^3 X_2), +(\pi X_2, \pi^4 X_2), +(\pi^2 X_2, \pi^5 X_2)) \rangle$$

by instantiating $X_2 \mapsto 1$.

Example 2 I

Let $*$ and $+$ be AC-function symbols and $\pi = (123456)$. Consider the solution:

$$\langle \emptyset, X \mapsto *(+(1,3,5),+(2,4,6)) \rangle$$

How can we inductively generate this solution?

Example 2 II

1. In the first iteration we may consider $m = 2$ and the permutation $\psi = (12)$. Then, we would instantiate $X \mapsto *(X_1, \pi X_1)$ and proceed to solve $\pi^2 X_1 \approx^? X_1$.

2. In the second iteration we may consider $m = 3$ and the permutation $\psi = (123)$. Our algorithm would instantiate $X_1 \mapsto +(X_2, \pi^2 X_2, \pi^4 X_2)$ and proceed to solve $(\pi^2)^3 X_2 \approx^? X_2$.

Example 2 III

3. In the third iteration, notice that $(\pi^2)^3 = \pi^6 = Id$. The solution to $\pi^6 X_2 \approx^? X_2$ would be $\langle \emptyset, Id \rangle$.

4. Plugging this value back, we would generate the solution

$$\langle \emptyset, X \mapsto *(+(X_2, \pi^2 X_2, \pi^4 X_2), +(\pi X_2, \pi^3 X_2, \pi^5 X_2) \rangle$$

Example 2 IV

The particular solution:

$$\langle \emptyset, X \mapsto *(+(1,3,5),+(2,4,6)) \rangle$$

can be obtained from

$$\langle \emptyset, X \mapsto *(+(X_2,\pi^2 X_2,\pi^4 X_2),+(\pi X_2,\pi^3 X_2,\pi^5 X_2) \rangle$$

by instantiating $X_2 \mapsto 1$.

A Modified Example 2 I

What happens if we change the previous example to consider $\pi = (123456)(7891011)$?

A Modified Example 2 II

In the first two steps the algorithm would proceed as in the previous example.

In the third, we would have the equation $\pi^6 X_2 \approx^? X_2$, where $\pi^6 = (7891011)$ and we would solve it by

$$\langle \{7, 8, 9, 10, 11\} \# X_2, Id \rangle$$

A Modified Example 2 III

Plugging this value back, we would get the solution:

$$\langle \{7, 8, 9, 10, 11\} \# X_2, X \mapsto *(+(X_2, \pi^2 X_2, \pi^4 X_2), +(\pi X_2, \pi^3 X_2, \pi^5 X_2)) \rangle$$

which is more general than

$$\langle \emptyset, X \mapsto *(+(1, 3, 5), +(2, 4, 6)) \rangle$$

by taking the instantiation $X_2 \mapsto 1$

Example 3 I

Let $*$ and $+$ be AC-function symbols and $\pi = (123456)(78)$.
Consider the solution:

$$\langle \emptyset, X \mapsto *(+(1,3,5,7),+(2,4,6,8)) \rangle$$

How can we inductively generate this solution?

Example 3 II

1. In the first iteration we may consider $m = 2$ and the permutation $\psi = (12)$. Then, we would instantiate $X \mapsto *(X_1, \pi X_1)$ and proceed to solve $\pi^2 X_1 \approx^? X_1$.

2. In the second iteration we may consider $m = 4$ and the permutation $\psi = (123)(4)$. Then, we would instantiate $X_1 \mapsto +(X_2, \pi^2 X_2, \pi^4 X_2, X_3)$ and proceed to solve $(\pi^2)^3 X_2 \approx^? X_2$ and $(\pi^2)^1 X_3 \approx^? X_3$.

Example 3 III

3. Since $\pi^6 = Id$, the solution to $(\pi^2)^3 X_2 \approx^? X_2$ is $\langle \emptyset, Id \rangle$.

4. One base solution to $\pi^2 X_3 \approx^? X_3$ is $\langle \{1,2,3,4,5,6\} \# X_3, Id \rangle$.

Example 3 IV

Plugging back the solutions we get

$$\langle \{1,2,3,4,5,6\} \# X_3, \\ X \mapsto *(+(X_2, \pi^2 X_2, \pi^4 X_2, X_3), +(\pi X_2, \pi^3 X_2, \pi^5 X_2, \pi X_3)) \rangle$$

which is actually more general than:

$$\langle \emptyset, X \mapsto *(+(1,3,5,7), +(2,4,6,8)) \rangle$$

since we can take the instantiation:

$$X_2 \mapsto 1, X_3 \mapsto 7$$

What happens with more than one fixed-point equation

If $P = \{\pi X \approx^? X, \rho X \approx^? X\}$, what do we do?

Idea: Follow the approach described in the FROCOS paper "On Solving Nominal Fixpoint Equations".

Frocos Approach - Notation

Let $\{\pi_i X \approx^? X\}$ be the unification problem we have to solve.

Frocos - General AC-Matcher

Definition 7 of the Frocos paper:

Definition 1

Let $t_1, \ldots, t_k$ be terms. We say that δ is a most general AC-matcher of the t_is if it is a most general AC-unifier of the problem $\{Z \approx^? t_i\}_{i=1,\ldots,k}$, where Z is a new variable.

Frocos - Generated Solutions

1. For each i, let $\langle \Gamma_i, X \mapsto t_i \rangle$ be an arbitrary solution (if exists any) to $\pi_i X \approx^? X$.
2. Find (if exists) the most general AC-matcher δ of the terms t_i. Consider X the new variable.
3. Given every
$$a\#Y \in \bigcup_{1 \leq i \leq k} \Gamma_i,$$
we see if there is some Γ such that $\Gamma \vdash a\#\delta Y$.
4. The solution is: $\langle \Gamma, \delta \rangle$.

PS: This is Definition 8 of the Frocos paper.

Alternative Approach - Adapting the Inductive Generation of Solutions

Let's say we want to solve $\{\pi X \approx^? X, \rho X \approx^? X\}$. As we commented on the last meeting, a possibility is to adapt our inductive thinking to handle more than one fixpoint equation. Let' say that we have a solution (Γ, σ) to both equations. Let's denote σX as t.

Alternative Approach - Base Cases

Atoms. The base case for atoms is still less general than the one for variables, so we would drop that.

Variables. We would output the solution

$$\langle dom(\pi)\#X \cup dom(\rho)\#X, Id \rangle$$

Alternative Approach - Inductive Cases

Syntactic Function. If $t = g(t_1, \ldots, t_m)$ we would try to find the solutions to $\{\pi X_i \approx^? X_i, \rho X_i \approx^? X_i\}$ for every i and then assemble them together as described for the syntactic function case where we only had one fixpoint equation.

Abstraction. Similar to the case where we only had one fixpoint equation.

Alternative Approach - AC Case I

If $t = f(t_1, \ldots, t_m)$ we have:

$$f(\pi t_1, \ldots, \pi t_m) \approx^? f(t_1, \ldots, t_m) \approx^? f(\rho t_1, \ldots, \rho t_m)$$

Alternative Approach - AC Case II

This case is more problematic because it is as if the equation $\pi X \approx^? X$ "forces" the instantiation:

$$X \mapsto f(\underbrace{X_1, \pi X_1, \ldots, \pi^{l_1-1} X_1}, \ldots, \underbrace{X_k, \pi X_k, \ldots, \pi^{l_k-1} X_k})$$

while the equation $\rho X \approx^? X$ "forces" the instantiation:

$$X \mapsto f(\underbrace{X'_1, \rho X'_1, \ldots, \rho^{l'_1-1} X'_1}, \ldots, \underbrace{X'_k, \rho X'_k, \ldots, \rho^{l'_k-1} X'_k})$$

Relating π and ρ in the AC case

Idea: A term t_k is associated with the moderated variable $\pi^{i_1} X_{i_2}$ **and** also with the moderated variable $\rho^{j_1} X'_{j_2}$ and hence we will have the equation $\pi^{i_1} X_{i_2} \approx^{?} \rho^{j_1} X'_{j_2}$.

Sketch of an Example I

Let $\sigma X = t = f(t_1, \ldots, t_6)$.
Consider that the permutation associated with π is $\psi_1 = (123)\ (456)$, i.e. the substitution associated is:

$$X \mapsto f(X_1, \pi X_1, \pi^2 X_1, X_2, \pi X_2, \pi^2 X_2).$$

Consider that the permutation associated with ρ is $\psi_2 = (12)\ (3456)$, i.e. the substitution associated is:

$$X \mapsto f(X_1', \pi X_1', X_2', \pi X_2', \pi^2 X_2', \pi^3 X_2')$$

Sketch of an Example II

The equations we have to solve are:

$$\pi^3 X_1 \approx^? X_1,\ \pi^3 X_2 \approx^? X_2$$
$$\rho^2 X_1' \approx^? X_1',\ \rho^4 X_2' \approx^? X_2'$$
$$X_1' \approx^? X_1,\ X_2' \approx^? \pi^2 X_1, \pi X_2' \approx^? X_2$$

Sketch of an Example III

Of course we start by instantiating the last ones:

$$X_1' \mapsto X_1$$
$$X_2' \mapsto \pi^2 X_1$$
$$X_2 \mapsto \pi^3 X_1$$

Sketch of an Example IV

And in the next iteration, the equations we will work on are:

$$\pi^3 X_1 \approx^? X_1,\ \rho^2 X_1 \approx^? X_1,\ \pi^{-2}\rho^4\pi^2 X_1 \approx^? X_1$$

and we have:

$$X \mapsto f(X_1, \pi X_1, \pi^2 X_1, \pi^3 X_1, \pi^4 X_1, \pi^5 X_1)$$

What about fixpoint equations with more than one variable?

If we have the equations $\pi_1 X \approx^? X$ and $\pi_2 Y \approx^? Y$ we can solve them separately obtaining solutions $(\Gamma_1, \{X \mapsto t\})$ and $(\Gamma_2, \{Y \mapsto s\})$ for the first and the second and then combine them obtaining the solution:

$$\langle \Gamma_1 \cup \Gamma_2, \{X \mapsto t,\ \ Y \mapsto s\} \rangle$$

Nice Triple

As we go applying the rules, the triple (Γ, σ, FP) maintain certain relations, which will be used in the proof of correctness and completeness. We collect those in the following definition:

Definition 2

We say that (Γ, σ, FP) is a nice triple if the following conditions are satisfied:

1. $Vars(FP) \cap dom(\sigma) = \emptyset$.
2. TO DO: I will add as we go advancing in the proofs of correctness and completeness.

Correctness

Theorem 3

Suppose that (Γ, σ, FP) *is a nice triple. If* (∇, δ) *is obtained from* (Γ, σ, FP) *after finitely many applications of the rules in* SOLVEFIXPOINT, *then:*

- $\nabla \vdash \delta(\pi_i X_i) \approx \delta X_i$ *for every* $\{\pi_i X_i \approx^? X_i\} \in FP$.
- $\nabla \vdash a\#\delta t$ *for every* $a\#_? t \in FP$.

Corollary 4

If $(\nabla, \delta) \in \text{SOLVEFIXPOINT}(\emptyset, Id, \{\pi X \approx^? X\})$ *then* $\nabla \vdash \delta(\pi X) \approx^? \delta X$.

Proof of Correctness

- It's in a separate file.
- Depends on the correctness of each rule. I proved for the all the cases of rules.

Completeness

TO DO: I will try to do it until next meeting.

Bounds in the Enumeration Procedure

We'll put a bound in the enumeration procedure, to obtain a terminating algorithm. We will bind by the depth of n_d and also by the arity of the flattened form of AC-functions m.