Formalising Nominal AC-Matching

XX Seminário Informal (, mas Formal!) 2023

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

1. Introduction

2. First Order AC-Unification
   Example of the AC Step for AC-Unification

3. The Nominal Setting

4. Adapting AC-Unification to the Nominal Setting

5. Nominal AC-Matching

6. More Details About Adapting to Nominal AC-Unification

7. Generating all Solutions to $\pi X \approx ? X$
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 $f(a, b)$. 
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 work in progress 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

- A Complete Unification Algorithm for Associative-Commutative Functions (Stickel'75)
- Associative Commutative Unification (Fages'84)
- A Certified AC-Matching Algorithm (Contejean'04) * Coq
- A Formally Verified Solver for Homogeneous Linear Diophantine Equations. (MPSS'18) * Isabelle/HoL
- Formalising Nominal AC-Unification (Work in Progress) (UNIF'19) * PVS
- Future Work: Nominal AC-Unification

A Certified Algorithm for AC-Unification (FSCD - 2022) * PVS
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>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$2X_1 + X_2 + X_3$</th>
<th>$2Y_1 + Y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Associating New Variables

Associate new variables with each solution.

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

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$2X_1 + X_2 + X_3$</th>
<th>$2Y_1 + Y_2$</th>
<th>New Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$Z_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$Z_2$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>$Z_3$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>$Z_4$</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>$Z_5$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>$Z_6$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>$Z_7$</td>
</tr>
</tbody>
</table>
Old and New Variables

Observing the previous Table, 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 \( f(Z_1, Z_4) \).
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:

$$
\begin{align*}
\sigma_1 &= \{ Y \rightarrow f(b, b), Z \rightarrow f(a, X, X) \} \\
\sigma_2 &= \{ Y \rightarrow f(Z_2, b, b), Z \rightarrow f(a, Z_2, X, X) \} \\
\sigma_3 &= \{ X \rightarrow b, Z \rightarrow f(a, Y) \} \\
\sigma_4 &= \{ X \rightarrow f(Z_6, b), Z \rightarrow f(a, Y, Z_6, Z_6) \}
\end{align*}
$$
Termination, Soundness and Completeness

We have proved termination, soundness and completeness:

- Termination - Hard
- Soundness - Easy
- Completeness - Hard
## Amount of Theorems and TCCs Proved

**Table:** Number of theorems and TCCs in each file.

<table>
<thead>
<tr>
<th>File</th>
<th>Theorems</th>
<th>TCCs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>unification_alg.pvs</code></td>
<td>10</td>
<td>19</td>
<td>29</td>
</tr>
<tr>
<td><code>termination_alg.pvs</code></td>
<td>80</td>
<td>35</td>
<td>115</td>
</tr>
<tr>
<td><code>rename_input.pvs</code></td>
<td>21</td>
<td>23</td>
<td>44</td>
</tr>
<tr>
<td><code>apply_ac_step.pvs</code></td>
<td>29</td>
<td>12</td>
<td>41</td>
</tr>
<tr>
<td><code>aux_unification.pvs</code></td>
<td>204</td>
<td>58</td>
<td>262</td>
</tr>
<tr>
<td><code>diophantine.pvs</code></td>
<td>73</td>
<td>44</td>
<td>117</td>
</tr>
<tr>
<td><code>unification.pvs</code></td>
<td>86</td>
<td>14</td>
<td>100</td>
</tr>
<tr>
<td><code>substitution.pvs</code></td>
<td>144</td>
<td>22</td>
<td>166</td>
</tr>
<tr>
<td><code>equality.pvs</code></td>
<td>67</td>
<td>18</td>
<td>85</td>
</tr>
<tr>
<td><code>terms.pvs</code></td>
<td>131</td>
<td>48</td>
<td>179</td>
</tr>
<tr>
<td><code>list.pvs</code></td>
<td>256</td>
<td>110</td>
<td>366</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1101</strong></td>
<td><strong>403</strong></td>
<td><strong>1504</strong></td>
</tr>
</tbody>
</table>
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.
The nominal setting extends first-order syntax, replacing the concept of syntactical equality by $\alpha$-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 $\mathbf{X} = \{X, Y, Z, \ldots\}$ and a set of atoms $\mathbf{A} = \{a, b, c, \ldots\}$.
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 \( \pi \) 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 \( \Delta \), \( \nabla \) or \( \Gamma \).
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 $\text{dom}(\pi) \# X$ 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 problem before and after are respectively:

\[ P = \{ f(X, W) \approx? f(\pi \cdot X, \pi \cdot Y) \} \]
\[ P_1 = \{ f(X_1, W_1) \approx? f(\pi \cdot X_1, \pi \cdot Y_1) \} \]
Focus on Nominal AC-Matching

Due to time constraints, we switched 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 that can be instantiated.
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.

- **$\alpha$-Equivalence** - By putting $\mathcal{X}$ as the set of variables that appear in the problem.
From unification to matching via $\mathcal{X}$ II

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

This approach could be used in future works to reason about nominal AC-unification, and it takes advantage from the fact that we already have a first-order AC-unification algorithm formalised.
Nominal AC-Matching

Things to Worry About:

- Does the matching problem “stays” a matching problem? - Done
- Termination - Almost Done
- Soundness and Completeness - TO DO
Does the matching problem “stays” a matching problem? I

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

But when we start introducing the new variables $Z_i$s, can we get a problem where an unprotected variable appears in the right-hand side?
Does the matching problem “stays” a matching problem?

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

Given a matching problem $P$, the idea is to use a lexicographic measure like

$$(\text{Vars}(P), \text{size}(P))$$

- $\text{Vars}(P)$ is the set of variables in the problem $P$.
- $\text{size}(P)$ is the multiset of the size of each equation $t_i \approx^? s_i \in P$. 
Let \( f(s_1, \ldots, s_m) \approx? f(t_1, \ldots, t_n) \) be the equation to which we apply the AC-step.

If after AC-step we do not instantiate any variable, then the equations after the AC-Step will be of the form \( t_i \approx? s_j \) and hence the size component of the lexicographic measure will decrease.

If we instantiate a variable, then the Vars component of the lexicographic measure will decrease.
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! Any comments/suggestions/doubts? ¹

¹to see more of my work, visit https://gabriel951.github.io/.


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\(^2\) is \(U_1 + U_2 = V_1 + V_2\) and the table of solutions is:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>New Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(Z_1)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>(W_1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(Y_1)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>(X_1)</td>
</tr>
</tbody>
</table>

\(^2\)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 AC Step

\[
\begin{align*}
\{ & X \approx? X_1, W \approx? Z_1, \pi \cdot X \approx? X_1, \pi \cdot Y \approx? Z_1 \} \\
\{ & X \approx? Y_1, W \approx? W_1, \pi \cdot X \approx? W_1, \pi \cdot Y \approx? Y_1 \} \\
\{ & X \approx? Y_1 + X_1, W \approx? W_1, \pi \cdot X \approx? W_1 + X_1, \pi \cdot Y \approx? Y_1 \} \\
\{ & X \approx? Y_1 + X_1, W \approx? Z_1, \pi \cdot X \approx? X_1, \pi \cdot Y \approx? Z_1 + Y_1 \} \\
\{ & X \approx? X_1, W \approx? Z_1 + W_1, \pi \cdot X \approx? W_1 + X_1, \pi \cdot Y \approx? Z_1 \} \\
\{ & X \approx? Y_1, W \approx? Z_1 + W_1, \pi \cdot X \approx? W_1, \pi \cdot Y \approx? Z_1 + Y_1 \} \\
\{ & X \approx? Y_1 + X_1, W \approx? Z_1 + W_1, \pi \cdot X \approx? W_1 + X_1, \pi \cdot Y \approx? Z_1 + Y_1 \}
\end{align*}
\]
After instantiating the variables

7 branches are generated:

\( B_1 \) - \( \{ \pi \cdot X \approx ? X \} \), \( \sigma = \{ W \mapsto \pi \cdot Y \} \)

\( B_2 \) - \( \sigma = \{ W \mapsto \pi^2 \cdot Y, X \mapsto \pi \cdot Y \} \)

\( B_3 \) - \( \{ f(\pi^2 \cdot Y, \pi \cdot X_1) \approx ? f(W, X_1) \} \), \( \sigma = \{ X \mapsto f(\pi \cdot Y, X_1) \} \)

\( B_4 \) - No solution

\( B_5 \) - No solution

\( B_6 \) - \( \sigma = \{ W \mapsto f(Z_1, \pi \cdot X), Y \mapsto f(\pi^{-1} \cdot Z_1, \pi^{-1} \cdot X) \} \)

\( B_7 \) - \( \{ f(\pi \cdot Y_1, \pi \cdot X_1) \approx ? f(W_1, X_1) \} \),

\[ \sigma = \{ X \mapsto f(Y_1, X_1), W \mapsto f(Z_1, W_1), Y \mapsto f(\pi^{-1} \cdot Z_1, \pi^{-1} \cdot Y_1) \} \]
Focusing on \textit{Branch7}, notice that the problem before the AC Step and the problem after the AC Step 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 \( Y = Y_0 \).
Let $k$ be the order of $\pi$. 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 $\sigma$ is of the form $\sigma_B y \sigma_{Bx_n} \ldots \sigma_{Bx_1}$, 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 $\delta$ is of the form $\delta' \delta_B y \delta_{Bx_m} \ldots \delta_{Bx_1}$, where $x_i$ is either 3 or 7 and $y$ is different than 3 or 7.
At the $i$-th iteration, the substitutions $\sigma_{B3}$ and $\sigma_{B7}$ differ only by the fact that $\sigma_{B7}$ 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 $\pi_1X, \ldots, \pi_nX$. 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 $f(X, Y)$. 
The optional argument notation let us write $\sigma_{Bx_i}$ 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.
Let's calculate $\sigma_{Bx_n} \ldots \sigma_{Bx_1}$ and $\delta_{Bx_m} \ldots \delta_{Bx_1}$ applied to $X_0$, $W_0$ and $Y_0$. 
\[ \sigma_{Bx_n} \ldots \sigma_{Bx_1} \quad \text{and} \quad \delta_{Bx_m} \ldots \delta_{Bx_1} \| \]

\[
X_0 \mapsto f(Y_1, X_1)
\]
\[
\mapsto f(\pi^{-1}Z_2^o, [\pi^{-1}, Id]Y_2, X_2)
\]
\[
\mapsto f(\pi^{-1}Z_2^o, [\pi^{-2}, \pi^{-1}]Z_3^o, [\pi^{-2}, \pi^{-1}, Id]Y_3, X_3)
\]
\[
\vdots
\]
\[
\mapsto f(\pi^{-1}Z_2^o, \ldots, [\pi^{-(n-1)}], \ldots, \pi^{-1}]Z_n^o, [\pi^{-(n-1)}], \ldots, Id]Y_n, X_n)
\]
σ_{Bx_n} \ldots \sigma_{Bx_1} \text{ and } \delta_{Bx_m} \ldots \delta_{Bx_1} \quad \text{III}

\begin{align*}
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{align*}
\[ \sigma_{Bx_n \ldots Bx_1} \text{ and } \delta_{Bx_m \ldots Bx_1} \quad \text{IV} \]

\[ Y_0 \mapsto f(\pi^{-1}Z_1^o, \pi^{-1}Y_1) \]

\[ \mapsto f(\pi^{-1}Z_1^o, \pi^{-2}Z_2^o, \pi^{-2}Y_2) \]

\[ \vdots \]

\[ \mapsto f(\pi^{-1}Z_1^o, \pi^{-2}Z_2^o, \ldots, \pi^{-n}Z_n^o, \pi^{-n}Y_n) \]
\( \sigma_{Bx_n} \ldots \sigma_{Bx_1} \) and \( \delta_{Bx_m} \ldots \delta_{Bx_1} \) 

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 $n \equiv m \pmod{k}$. Consider the substitution $\sigma^* = \sigma_{Bx_n} \ldots \sigma_{Bx_1}$, where

$$\sigma_{Bx_i} = \begin{cases} 
\sigma_{B7}, & \text{if } i \leq k \text{ and } \\
\exists j : j \equiv i \pmod{k} \text{ and } Z_j \in \text{Args}(\delta_{Bx_m} \ldots \delta_{Bx_1} W_0) & \\
\sigma_{B3}, & \text{otherwise}
\end{cases}$$
We can find $\lambda$ such that $\delta_{Bx_m} \ldots \delta_{Bx_1} = \lambda \sigma^*$. Define $\lambda$ by:

- If $i \leq k$ and $\exists j : j \equiv i \pmod{k}$ 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 $\text{Args}(\delta_{Bx_m} \ldots \delta_{Bx_1} W_0)$

- Otherwise, $\lambda Z_i \mapsto Z_i$
\[ \sigma^* \leq \delta_{Bx_m} \ldots \delta_{Bx_1} \ | |

- \lambda Y_n \leftrightarrow Y_m
- \lambda W_n \leftrightarrow W_m
Given a variable $Z_j$, let $i_j$ be the index such that $i_j \leq k$ and $i_j \equiv j \pmod{k}$. Then,

$$\lambda X_n \mapsto f([\pi^{-i_{k+1}}, \ldots, \pi^{-k}]Z_{k+1}^o, \ldots, [\pi^{-i_n}, \ldots, \pi^{-(n-1)}]Z_n^o, [\pi^{-n}, \ldots, \pi^{-(m-1)}]Y_m, X_m)$$
A set of triples is enough

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 $n \leq 2k$. 
If $\delta$ is of the form $\delta' \delta_{B_{y_i}} \delta_{B_{x_m}} \ldots \delta_{B_{x_1}}$, then the triple output by the algorithm that we are looking for would be $(\emptyset, \sigma_{B_{y_i}} \sigma^*, FP_{B_{y_i}})$, where $FP_{B_{y_i}}$ 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 $\pi_1 X \in \text{ Args}(t)$ and $\pi_2 X \in \text{ Args}(s)$, is a set of triples always enough? Can we generalise the argument we used for $f(X, W) \approx f(\pi X, \pi Y)$?
We may not get exactly a loop after applying the AC Step and after we instantiate the variables. 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 $\pi X \approx? X$?
Solving $\pi X \approx? X$ is equivalent to finding all the terms $t$ such that there is a context $\Gamma$ such that $\Gamma \vdash \pi t \approx? t$. 
A trivial procedure

Generate every term \( t \) and then find (if possible) the minimal context \( \nabla \) 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.
The enumeration procedure will be given as a set of non-deterministic rules, that operate on triples of the form \((\Gamma, \sigma, FP)\), 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 \((\emptyset, Id, \{\pi X \approx? X\})\).
Rules of the enumeration procedure:

- (Var)
- (Func)
- (Abs a) and (Abs b)
- (AC Func)
- Old rules for solving freshness problems
- (Term)
The freshness problems are introduced by rule \((\text{Abs } b)\).

As we go applying the enumeration rules, no variable \(X\) appear in more than one fixpoint equation.
(Var) rule:

\[(\Gamma, \sigma, \{\pi X \simeq^? X\} \cup FP) \xrightarrow{\text{Var}} (\Gamma \cup dom(\pi)\#X, \sigma, FP)\]
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:

\[
(\Gamma, \sigma, \{ \pi X \approx ? X \} \cup FP) \xrightarrow{\text{Func}} (\Gamma, \sigma' \sigma, \{ \pi X_1 \approx ? X_1, \ldots, \pi X_m \approx ? X_m \} \cup \sigma' FP)
\]
Abstraction Rule - First case

Let \( a \not\in \text{dom}(\pi) \). Let \( \sigma' = \{ X \mapsto [a]X_1 \} \), where \( X_1 \) is a new variable.

The \((\text{Abs } a)\) rule:

\[
(\Gamma, \sigma, \{ \pi X \approx? X \} \cup FP) \xrightarrow{\text{Abs } a} (\Gamma, \sigma' \sigma, \{ \pi X_1 \approx? X_1 \} \cup \sigma' FP)
\]
Abstraction Rule - Second case

Let \( a \in \text{dom}(\pi) \). Let \( \pi' = (a \pi a) \pi \) and let \( \sigma' = \{ X \mapsto [a]X_1 \} \), where \( X_1 \) is a **new** variable.

The \((\text{Abs } b)\) rule:

\[
(\Gamma, \sigma, \{ \pi X \approx ? X \} \cup FP) \xrightarrow{\text{Abs } b} \\
(\Gamma, \sigma' \sigma, \{ \pi'X_1 \approx ? X_1 \} \cup \sigma'FP \cup \{ a\#?\pi X_1 \})
\]
AC Function rule I

Let $m$ be an arbitrary number and let $\psi$ 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 $\sigma'$ be:

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

The (AC Func) rule is:

\[(\Gamma, \sigma, \{\pi X \approx? X\} \cup FP) \xrightarrow{ACFunc} (\Gamma, \sigma', \{\pi_1 X_1 \approx? X_1, \ldots, \pi_k X_k \approx? X_k\} \cup \sigma' FP)\]
Termination rule

\[(\text{Term}) \text{ rule:} \]

\[(\Gamma, \sigma, \emptyset) \xrightarrow{\text{Term}} (\Gamma, \sigma)\]
Solution when $t$ is an Atom

A solution is when $t = \sigma X$ is an atom $a_i \not\in \text{dom}(\pi)$:

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

Notice that this solution, however, is less general than $\langle \text{dom}(\pi)\#X, \text{Id} \rangle$ if we consider the substitution $\sigma' = X \mapsto a_i$. Therefore, there is no need for a rule for atoms.
Let $\ast$ and $+$ be AC-function symbols and $\pi = (123456)$. Consider the solution:

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

How can we inductively generate this solution?
Example 1 II

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

2. In the second rule application 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 \( (\pi^3)^2 X_2 \approx? X_2 \).
3. In the third rule application, 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

\[
\langle \emptyset, X \mapsto \ast (+ (X_2, \pi^3 X_2), + (\pi X_2, \pi^4 X_2), + (\pi^2 X_2, \pi^5 X_2)) \rangle
\]
The particular solution:

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

can be obtained from:

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

by instantiating $$X_2 \mapsto 1.$$
Example 2

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 rule application 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 rule application 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 rule application, notice that $(\pi^2)^3 = \pi^6 = \text{Id}$. The solution to $\pi^6 X_2 \approx? X_2$ would be $\langle \emptyset, \text{Id} \rangle$.

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

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

The particular solution:

\[ \langle \emptyset, X \mapsto \ast((1, 3, 5), (2, 4, 6)) \rangle \]

can be obtained from

\[ \langle \emptyset, X \mapsto \ast((X_2, \pi^2 X_2, \pi^4 X_2), (\pi X_2, \pi^3 X_2, \pi^5 X_2)) \rangle \]

by instantiating \( X_2 \mapsto 1 \).
A Modified Example 2

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

$$\langle\{7, 8, 9, 10, 11\} \# X_2, \text{Id}\rangle$$
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 3.1

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

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

How can we inductively generate this solution?
1. In the first rule application we may consider $m = 2$ and the permutation $\psi = (12)$. Then, we would instantiate $X \mapsto \ast (X_1, \pi X_1)$ and proceed to solve $\pi^2 X_1 \approx ? X_1$.

2. In the second rule application 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 $(\pi^2)^1 X_3 \approx ? X_3$. 
3. Since $\pi^6 = Id$, the solution to $(\pi^2)^3X_2 \approx ? X_2$ is $\langle \emptyset, Id \rangle$.

4. One base solution to $\pi^2X_3 \approx ? X_3$ 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 \( \{\pi; X \approx? X\} \) be the unification problem we have to solve.
Definition 1 of the Frocos paper:

**Definition 1**
Let $t_1, \ldots, t_k$ be terms. We say that $\delta$ is a most general AC-matcher of the $t_i$s 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.
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 \( \delta \) 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 \( \Gamma \) such that \( \Gamma \vdash a\# \delta Y \).

4. The solution is: \( \langle \Gamma, \delta \rangle \).

PS: This is Definition 8 of the Frocos paper.
Let's say we want to solve \( \{ \pi X \approx ? X, \rho X \approx ? X \} \). One possibility is to adapt our inductive thinking to handle more than one fixpoint equation. Let’s say that we have a solution \((\Gamma, \sigma)\) to both equations. Let’s denote \( \sigma X \) as \( t \).
**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

$$\langle \text{dom}(\pi)\#X \cup \text{dom}(\rho)\#X, \text{Id} \rangle$$
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.
If \( t = f(t_1, \ldots, t_m) \) we have:

\[
\begin{align*}
    f(\pi t_1, \ldots, \pi t_m) & \approx ? f(t_1, \ldots, t_m) \approx ? f(\rho t_1, \ldots, \rho t_m)
\end{align*}
\]
This case is more problematic because it is as if the equation $\pi X \approx ? X$ “forces” the instantiation:

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

while the equation $\rho X \approx ? X$ “forces” the instantiation:

$$X \mapsto f(X'_1, \rho X'_1, \ldots, \rho^{l'_1-1}X'_1, \ldots, X'_k, \rho X'_k, \ldots, \rho^{l'_k-1}X'_k)$$
Relating $\pi$ and $\rho$ 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 \( \pi \) 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 \( \rho \) 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:

\[ \pi^3 X_1 \approx? X_1, \, \pi^3 X_2 \approx? X_2 \]
\[ \rho^2 X'_1 \approx? X'_1, \, \rho^4 X'_2 \approx? X'_2 \]
\[ X'_1 \approx? X_1, \, X'_2 \approx? \pi^2 X_1, \pi X'_2 \approx? X_2 \]
Of course we start by instantiating the last ones:

\[ X'_1 \mapsto X_1 \]
\[ X'_2 \mapsto \pi^2 X_1 \]
\[ X_2 \mapsto \pi^3 X_1 \]
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:

$$\langle \Gamma_1 \cup \Gamma_2, \{X \mapsto t, \ Y \mapsto s\} \rangle$$
As we go applying the rules, the triple \((\Gamma, \sigma, FP)\) 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 \((\Gamma, \sigma, FP)\) is a nice triple if the following conditions are satisfied:

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

Theorem 3
Suppose that \((\Gamma, \sigma, FP)\) is a nice triple. If \((\nabla, \delta)\) is obtained from \((\Gamma, \sigma, FP)\) after finitely many applications of the rules in \textsc{solveFixpoint}, then:

- \(\nabla \vdash \delta(\pi_iX_i) \approx \delta X_i\) for every \(\{\pi_iX_i \approx ? X_i\} \in FP\).
- \(\nabla \vdash a\#\delta t\) for every \(a\#? t \in FP\).

Corollary 4
If \((\nabla, \delta) \in \textsc{solveFixpoint}(\emptyset, \text{ld}, \{\pi X \approx ? X\})\) 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
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 $m$. 