# Nominal Anti-Unification

Alexander Baumgartner*, Temur Kutsia*, Jordi Levy†, and Mateu Villaret‡
*Research Institute for Symbolic Computation, Johannes Kepler University Linz, Austria
{abaumgar, kutsia}@risc.jku.at
†Artificial Intelligence Research Institute, Spanish Council for Scientific Research, Barcelona, Spain
levy@iiia.csic.es
‡Departament d'Informàtica i Matemàtica Aplicada, Universitat de Girona, Spain
villaret@ima.udg.edu

***Abstract*—We study nominal anti-unification, which is concerned with computing least general generalizations for given terms-in-context. In general, the problem does not have a least general solution, but if the set of atoms permitted in generalizations is finite, then there exists a least general generalization which is unique modulo variable renaming and $\alpha$-equivalence. We present an algorithm that computes it. The algorithm relies on a subalgorithm that constructively decides equivariance between two terms-in-context. We prove soundness and completeness properties of both algorithms and analyze their complexity.**

## I. Introduction

Binders are very common in computer science, logic, mathematics, linguistics. Functional abstraction $\lambda$, universal quantifier $\forall$, limit $\lim$, integral $\int$ are some well-known examples of binders. To formally represent and study systems with binding, Pitts and Gabbay [10]–[12] introduced nominal techniques, based on the idea to give explicit names to bound entities. It makes a syntactic distinction between *atoms,* which can be bound, and *variables,* which can be substituted. This approach led to the development of the theory of nominal sets, nominal logic, nominal algebra, nominal rewriting, nominal logic programming, etc.

When dealing with names explicitly, one needs to distinguish which of them are fresh for a term in order, e.g., to do $\alpha$-conversion. Such constraints can be specified in the *freshness context.*

Renaming should be performed explicitly as well. It is done via permutation of atoms. When such a permutation is applied to a term, it renames the involved atoms and gets suspended at variables. Once a variable are instantiated, the suspended permutation is applied to the instance.

Equation solving between nominal terms (maybe together with freshness constraints) has been investigated by several authors, who designed and analyzed algorithms for nominal unification [3], [4], [15], [24], nominal matching [5], equivariant unification [6], permissive nominal unification [8]. However, in contrast to unification, its dual problem, anti-unification, has not been studied for nominal terms previously. In [17], it is referred to as "the as-of-yet undiscovered nominal anti-unification", which "could form a fundamental component of a refactoring tool" for $\alpha$Prolog [7] programs.

The anti-unification problem for two terms $t_1$ and $t_2$ is concerned with finding a term $t$ that is more general than the original ones, i.e., $t_1$ and $t_2$ should be substitutive instances of $t$. The interesting generalizations are the least general ones, which retain the common structure of $t_1$ and $t_2$ as much as possible. Plotkin [19] and Reynolds [21] initiated research on anti-unification in the 1970s, developing generalization algorithms for first-order terms. Since then, anti-unification has been studied in various theories, including some of those with binding constructs: calculus of constructions [18], $M\lambda$ [9], second-order lambda calculus with type variables [16], simply-typed lambda calculus where generalizations are higher-order patterns [2], just to name a few.

The problem we address in this paper is to compute generalizations for nominal terms. More precisely, we consider this problem for nominal *terms-in-context*, which are pairs of a freshness context and a nominal term, aiming at computing their least general generalizations (lgg). However, it turned out that without a restriction, there is no lgg for terms-in-context, in general. Even more, a *minimal* complete set of generalizations does not exist. This is in sharp contrast with the related problem of anti-unification for higher-order patterns, which always have a single lgg [2].

Therefore we restrict the set atoms which are permitted in generalizations to be finite. In this case, there exists a single lgg (modulo $\alpha$-equivalence and variable renaming) for terms-in-context and we design an algorithm to compute it.

There is a close relation between nominal and higher-order pattern unification: One can be translated into the other by the solution-preserving translation defined in [15]. We show that for anti-unification, this method, in general, is not applicable. Even if one finds conditions under which such a translation-based approach to anti-unification works, due to complexity reasons it is still better to use the direct nominal anti-unification algorithm developed in this paper.

Computation of nominal lgg's requires to solve the equivariance problem: Given two terms $s_1$ and $s_2$, find a permutation of atoms which, when applied to $s_1$, makes it $\alpha$-equivalent to $s_2$ (under the given freshness context). This is necessary to guarantee that the computed generalization is *least* general. For instance, if the given terms are $s_1 = f(a, b)$ and $s_2 = f(b, a)$, where $a, b$ are atoms, the freshness context is empty, and the atoms permitted in the generalization are $a, b,$ and $c$, then the term-in-context $\langle\{c\#X, c\#Y\}, f(X, Y)\rangle$ generalizes $\langle\emptyset, s_1\rangle$ and $\langle\emptyset, s_2\rangle$, but it is not their lgg. To compute the

latter, we need to reflect the fact that generalizations of the atoms are related to each other: One can be obtained from the other by swapping $a$ and $b$. This leads to an lgg $\langle\{c\#X\}, f(X,(a\,b)\cdot X)\rangle$. To compute the permutation $(a\,b)$, an equivariance problem should be solved. We develop an algorithm for equivariance problems, which computes the justifying permutation if the input terms are equivariant, and fails otherwise.

The above mentioned software code refactoring is not the only application of anti-unification. Various variants of this technique, such as first-order, higher-order, or equational anti-unification have been used in inductive logic programming, logical and relational learning [20], reasoning by analogy [13], program synthesis [22], program verification [16], etc. Nominal anti-unification can, hopefully, contribute in solving similar problems in nominal setting.

In this paper, we mainly follow the notation from [15]. Long proofs are put in the appendix.

### *Contributions*

Contributions of this paper can be summarized as follows:

- The lattice structure of nominal terms-in-context is described, where meet represents to their lgg and join, when it exists, corresponds to their most general common instance. The latter is computed by nominal unification, reformulated for terms-in-context.
- A nominal anti-unification algorithm $\mathfrak{N}$ is designed. It accepts two terms-in-context and computes their lgg, a term-in-context, under the condition that the set of atoms permitted in generalizations is finite.
- Termination, soundness, and completeness theorems for $\mathfrak{N}$ are proved.
- Uniqueness of computed lgg modulo $\alpha$-equivalence and variable renaming is proved.
- A constructive decision algorithm for equivariance $\mathfrak{E}$ is designed. It decides whether two terms are equivariant with respect to a given freshness context and if they are, returns a justifying permutation. Otherwise it reports failure.
- Termination, soundness, and completeness theorems for $\mathfrak{E}$ are proved.
- Time and space complexity analysis of both algorithms is performed. The results are given in the table below:

| Algorithm | Time | Space |
|---|---|---|
| Anti-unification | $O(n^4)$ | $O(n^2)$ |
| Equivariance | $O(n^2)$ | $O(n^2)$ |

- Both algorithms have been implemented and are freely available. The implementation of $\mathfrak{N}$ can be accessed at:
  www.risc.jku.at/projects/stout/software/nau.php.
  The equivariance algorithm $\mathfrak{E}$ is a part of $\mathfrak{N}$, but one can also access it separately at:
  www.risc.jku.at/projects/stout/software/nequiv.php.

## II. Nominal Terms

Nominal terms contain *variables* and *atoms.* Variables can be instantiated and atoms can be bound. In *nominal signatures* we have *sorts of atoms* (typically $\nu$) and *sorts of data* (typically $\delta$) as disjoint sets. *Atoms* (typically $a, b, \ldots$) have one of the sorts of atoms. *Variables* (typically $X, Y, \ldots$) have a sort of atom or a sort of data, i.e. of the form $\nu \mid \delta$. Nominal function symbols (typically $f, g, \ldots$) have an arity of the form $\tau_1 \times \cdots \times \tau_n \to \delta$, where $\delta$ is a sort of data and $\tau_i$ are sorts given by the grammar $\tau ::= \nu \mid \delta \mid \langle\nu\rangle\tau$ . Abstractions have sorts of the form $\langle\nu\rangle\tau$.

A *swapping* $(a\,b)$ is a pair of atoms of the same sort. A *permutation* is a (possibly empty) sequence of swappings. We use upright Greek letters (e.g., $\pi, \rho$) to denote permutations.

*Nominal terms* (typically $t, s, u, r, q, \ldots$) are given by the grammar:

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

where $f$ is an $n$-ary function symbol, $a$ is an atom, $\pi$ is a permutation, and $X$ is a variable. They are called respectively *application, atom, abstraction, and suspension.*

The effect of a swapping over an atom is defined by $(a\,b)\cdot a = b$ and $(a\,b)\cdot b = a$ and $(a\,b)\cdot c = c$, when $c \notin \{a, b\}$. For the rest of terms the extension is straightforward, in particular, for abstractions $(a\,b)\cdot(c.t) = ((a\,b)\cdot c)\,.\,((a\,b)\cdot t)$. The effect of a permutation is defined by $(a_1\,b_1)\ldots(a_n\,b_n)\cdot t = (a_1\,b_1)\cdot((a_2\,b_2)\ldots(a_n\,b_n)\cdot t)$. The *inverse* of a permutation $\pi = (a_1\,b_1)\ldots(a_n\,b_n)$ is the permutation $(a_n\,b_n)\ldots(a_1\,b_1)$, denoted by $\pi^{-1}$. The empty permutation is denoted by $Id$.

The effect of the empty permutation is $Id\cdot t = t$. We extend it to suspensions and write $X$ as the shortcut of $Id\cdot X$.

The set of variables of a term $t$ is denoted by $\mathrm{Vars}(t)$. A term $t$ is called *ground* if $\mathrm{Vars}(t) = \emptyset$. The set of *atoms* of a term $t$ or a permutation $\pi$ is the set of all atoms which appear in it and is denoted by $\mathrm{Atoms}(t)$, $\mathrm{Atoms}(\pi)$ respectively. We write $\mathrm{Atoms}(\pi_1, \pi_2)$ for the set $\mathrm{Atoms}(\pi_1) \cup \mathrm{Atoms}(\pi_2)$.

The *length* of a term $t$ is the number of all appearances of atoms, variables, and function symbols in it and we denote it by $\|t\|$, e.g., $\|f(a.(a\,b)X, X, Y)\| = 7$. The number of variable occurrences in a term $t$ is denoted by $\|t\|_{\mathrm{Vars}}$, e.g., $\|f(a.(a\,b)X, X, Y)\|_{\mathrm{Vars}} = 3$.

*Positions* in terms are defined with respect to their tree representation in the usual way, as strings of integers. For instance, the tree form of the term $f(a.b.g((a\,b)\cdot X, a), h(c))$, and the corresponding positions are shown in Fig. 1. The symbol $f$ stands in the position $\epsilon$ (the empty sequence). The suspension is put in one node of the tree, at the position 1.1.1.1. The abstraction operator and the corresponding bound atom together occupy one node as well. For any term $t$, $t|_{\mathsf{p}}$

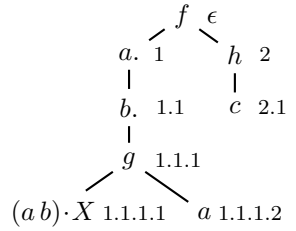


Fig. 1. The tree form and positions of $f(a.b.g((a\,b)\cdot X, a), h(c))$.

denotes the *subterm of* $t$ *at position* $\mathsf{p}$. For instance, $f(a.b.g((a\,b)\cdot X, a), h(c))|_{1.1} = b.g((a\,b)\cdot X, a)$.

The *path to a position* in a term is defined as the sequence of expressions from the root to the node at that position (not including) in the tree form of the term, e.g., the path to the position 1.1.1.2 in $f(a.b.g((a\,b)\cdot X, a), h(c))$ is $f, a., b., g$. Since suspensions are always in leaves, they never appear in a path.

Every permutation $\pi$ naturally defines a bijective function from the set of atoms to the sets of atoms, that we will also represent as $\pi$. Suspensions are uses of variables with a permutation of atoms waiting to be applied once the variable is instantiated. Occurrences of an atom $a$ are said to be bound if they are in the scope of an abstraction of $a$, otherwise are said to be free. We denote by $\mathrm{FA}(t)$ the set of all atoms which occur freely in $t$: $\mathrm{FA}(f(t_1,\ldots,t_n)) = \bigcup_{i=1}^{n} \mathrm{FA}(t_i)$, $\mathrm{FA}(a) = \{a\}$, $\mathrm{FA}(a.t) = \mathrm{FA}(t)\setminus\{a\}$, and $\mathrm{FA}(\pi\cdot X) = \mathrm{Atoms}(\pi)$. $\mathrm{FA}^{\text{-s}}(t)$ is the set of all atoms which occur freely in $t$ ignoring suspensions: $\mathrm{FA}^{\text{-s}}(f(t_1,\ldots,t_n)) = \bigcup_{i=1}^{n} \mathrm{FA}^{\text{-s}}(t_i)$, $\mathrm{FA}^{\text{-s}}(a) = \{a\}$, $\mathrm{FA}^{\text{-s}}(a.t) = \mathrm{FA}^{\text{-s}}(t) \setminus \{a\}$, and $\mathrm{FA}^{\text{-s}}(\pi\cdot X) = \emptyset$.

The head of a term $t$, denoted $\mathrm{Head}(t)$, is defined as: $\mathrm{Head}(f(t_1,\ldots,t_n)) = f$, $\mathrm{Head}(a) = a$, $\mathrm{Head}(a.t) = .$, and $\mathrm{Head}(\pi\cdot X) = X$.

Substitutions are defined in the standard way, as a mapping from variables to terms, and their application allows atom capture, for instance, $a.X\{X \mapsto a\} = a.a$. We use Greek letters $\sigma, \vartheta, \varphi$ to denote substitutions. The identity substitution is denoted by $\varepsilon$. The notions of substitution *domain* and *range* are also standard and are denoted, respectively, by $\mathrm{Dom}$ and $\mathrm{Ran}$.

A *freshness constraint* is a pair of the form $a\#X$ stating that the instantiation of $X$ cannot contain free occurrences of $a$. A *freshness context* is a finite set of freshness constraints. We will use $\nabla$ and $\Gamma$ to denote freshness contexts. $\mathrm{Vars}(\nabla)$ and $\mathrm{Atoms}(\nabla)$ denote respectively the set of variables and atoms of $\nabla$.

We say that a substitution $\sigma$ *respects* a freshness constraint $\nabla$, if for all $X$, $\mathrm{FA}^{\text{-s}}(X\sigma) \cap \{a \mid a\#X \in \nabla\} = \emptyset$.

The predicate $\approx$, which stands for $\alpha$-equivalence between terms, was defined in [23], [24] by means of the following theory:

$$\frac{}{\nabla \vdash a \approx a}(\approx\text{-atom}) \qquad \frac{\nabla \vdash t \approx t'}{\nabla \vdash a.t \approx a.t'}(\approx\text{-abs-1})$$

$$\frac{a \neq a' \quad \nabla \vdash t \approx (a\,a')\cdot t' \quad \nabla \vdash a\#t'}{\nabla \vdash a.t \approx a'.t'}(\approx\text{-abs-2})$$

$$\frac{a\#X \in \nabla \text{ for all } a \text{ such that } \pi\cdot a \neq \pi'\cdot a}{\nabla \vdash \pi\cdot X \approx \pi'\cdot X}(\approx\text{-susp.})$$

$$\frac{\nabla \vdash t_1 \approx t_1' \quad \cdots \quad \nabla \vdash t_n \approx t_n'}{\nabla \vdash f(t_1,\ldots t_n) \approx f(t_1',\ldots,t_n')}(\approx\text{-application})$$

where the freshness predicate $\#$ is defined by

$$\frac{a \neq a'}{\nabla \vdash a\#a'}(\#\text{-atom}) \qquad \frac{(\pi^{-1}\cdot a\#X) \in \nabla}{\nabla \vdash a\#\pi\cdot X}(\#\text{-susp.})$$

$$\frac{\nabla \vdash a\#t_1 \quad \cdots \quad \nabla \vdash a\#t_n}{\nabla \vdash a\#f(t_1,\ldots t_n)}(\#\text{-application})$$

$$\frac{}{\nabla \vdash a\#a.t}(\#\text{-abst-1}) \qquad \frac{a \neq a' \quad \nabla \vdash a\#t}{\nabla \vdash a\#a'.t}(\#\text{-abst-2})$$

Their intended meanings are:

1) $\nabla \vdash a\#t$ holds, if for every substitution $\sigma$ such that $t\sigma$ is a ground term and $\sigma$ respects the freshness context $\nabla$, we have $a$ is not free in $t\sigma$;
2) $\nabla \vdash t \approx u$ holds, if for every substitution $\sigma$ such that $t\sigma$ and $u\sigma$ are ground terms and $\sigma$ respects the freshness context $\nabla$, $t\sigma$ and $u\sigma$ are $\alpha$-equivalent.

Based on the definition of the freshness predicate, we can design an algorithm which solves the following problem: Given a set of pairs $\{a_1\#t_1,\ldots,a_n\#t_n\}$, compute a *minimal* (with respect to $\subseteq$) freshness context $\nabla$ such that $\nabla \vdash a_1\#t_1,\ldots,\nabla \vdash a_n\#t_n$. Such a $\nabla$ may or may not exist, and the algorithm should detect it.

We give a rule-based description of the algorithm, which we call FC for it is supposed to compute a freshness context. The rules operate on pairs $F;\nabla$, where $F$ is a set of atomic freshness formulas of the form $a\#t$, and $\nabla$ is a freshness context. $\uplus$ stands for disjoint union. The rules are the following:

Del-FC: **Delete in** FC

$\{a\#b\}\uplus F;\ \nabla \Longrightarrow F;\ \nabla, \qquad$ if $a \neq b$.

Abs-FC1: **Abstraction in** FC **1**

$\{a\#a.t\}\uplus F;\ \nabla \Longrightarrow F;\ \nabla$.

Abs-FC2: **Abstraction in** FC **2**

$\{a\#b.t\}\uplus F;\ \nabla \Longrightarrow \{a\#t\} \cup F;\ \nabla, \qquad$ if $a \neq b$.

Dec-FC: **Decomposition in** FC

$\{a\#f(t_1,\ldots,t_n)\}\uplus F;\ \nabla \Longrightarrow$
$\{a\#t_1,\ldots,a\#t_n\} \cup F;\ \nabla$.

Sus-FC: **Suspension in** FC

$\{a\#\pi\cdot X\}\uplus F;\ \nabla \Longrightarrow F;\ \{\pi^{-1}\cdot a\#X\} \cup \nabla$.

To compute a minimal freshness context which justifies the atomic freshness formulas $a_1\#t_1,\ldots,a_n\#t_n$, we start with $\{a_1\#t_1,\ldots,a_n\#t_n\};\emptyset$ and apply the rules of FC as long as possible. It is easy to see that the algorithm terminates. The system to which no rule applies has either the form $\emptyset;\nabla$ or $\{a\#a\} \cup F;\nabla$, where $\nabla$ is a freshness context. In the former case we say that the algorithm succeeds and computes $\nabla$, writing this fact as $\mathsf{FC}(\{a_1\#t_1,\ldots,a_n\#t_n\}) = \nabla$. In the latter case we say that FC fails and write $\mathsf{FC}(\{a_1\#t_1,\ldots,a_n\#t_n\}) = \bot$.

**Theorem 1.** *Let $F$ be a set of atomic freshness formulas and $\nabla$ be a freshness context. Then $\mathsf{FC}(F) \subseteq \nabla$ iff $\nabla \vdash a\#t$ for all $a\#t \in F$.*

*Proof:* By the structural induction over $t$, exploiting the similarity between the rules of FC and the definition of the freshness predicate $\#$. ∎

**Corollary 1.** *$\mathsf{FC}(F) = \bot$ iff there is no freshness context that would justify all formulas in $F$.*

Given a freshness context $\nabla$ and a substitution $\sigma$, we define $\nabla\sigma = \mathsf{FC}(\{a\#X\sigma \mid a\#X \in \nabla\})$. The following lemma is straightforward:

**Lemma 1.** *$\sigma$ respects $\nabla$ iff $\nabla\sigma \neq \bot$.*

When $\nabla\sigma \neq \bot$, we call $\nabla\sigma$ the *instance* of $\nabla$ under $\sigma$.

It is not hard to see that (a) if $\sigma$ respects $\nabla$, then $\sigma$ respects any $\nabla' \subseteq \nabla$, and (b) if $\sigma$ respects $\nabla$ and $\vartheta$ respects $\nabla\sigma$, then $\sigma\vartheta$ respects $\nabla$ and $(\nabla\sigma)\vartheta = \nabla(\sigma\vartheta)$.

A *term-in-context* is a pair $\langle \nabla, t\rangle$ of a freshness context and a term.

We say that a term-in-context $\langle \nabla_1, t_1\rangle$ is *more general* than a term-in-context $\langle \nabla_2, t_2\rangle$, written $\langle \nabla_1, t_1\rangle \preceq \langle \nabla_2, t_2\rangle$, if there exists a substitution $\sigma$, which respects $\nabla_1$, such that $\nabla_1\sigma \subseteq \nabla_2$ and $\nabla_2 \vdash t_1\sigma \approx t_2$.

We write $\nabla \vdash t_1 \preceq t_2$ if there exists a substitution $\sigma$ such that $\nabla \vdash t_1\sigma \approx t_2$.

Two terms-in-context $p_1$ and $p_2$ are *equivalent* (or *equi-general*), written $p_1 \simeq p_2$, iff $p_1 \preceq p_2$ and $p_2 \preceq p_1$. The strict part of $\preceq$ is denoted by $\prec$, i.e., $p_1 \prec p_2$ iff $p_1 \preceq p_2$ and not $p_2 \preceq p_1$. We also write $\nabla \vdash t_1 \simeq t_2$ iff $\nabla \vdash t_1 \preceq t_2$ and $\nabla \vdash t_2 \preceq t_1$.

**Example 1.** We give some examples to demonstrate the relations we have just defined:

- $\langle\{a\#X\}, f(a)\rangle \simeq \langle\emptyset, f(a)\rangle$. We can use $\{X \mapsto b\}$ for the substitution applied to the first pair.
- $\langle\emptyset, f(X)\rangle \preceq \langle\{a\#X\}, f(X)\rangle$ (with $\sigma = \varepsilon$), but not $\langle\{a\#X\}, f(X)\rangle \preceq \langle\emptyset, f(X)\rangle$.
- $\langle\emptyset, f(X)\rangle \preceq \langle\{a\#Y\}, f(Y)\rangle$ with $\sigma = \{X \mapsto Y\}$.
- $\langle\{a\#X\}, f(X)\rangle \not\preceq \langle\emptyset, f(Y)\rangle$, because in order to satisfy $\{a\#X\}\sigma \subseteq \emptyset$, the substitution $\sigma$ should map $X$ to a term $t$ which contains neither $a$ (freely) nor variables. But then $\emptyset \vdash f(t) \approx f(Y)$ does not hold. Hence, together with the previous example, we get $\langle\emptyset, f(Y)\rangle \prec \langle\{a\#X\}, f(X)\rangle$.
- $\langle\{a\#X\}, f(X)\rangle \not\preceq \langle\{a\#X\}, f(a)\rangle$. Notice that $\sigma = \{X \mapsto a\}$ does not respect $\{a\#X\}$.
- $\langle\{b\#X\}, (a\,b)\cdot X\rangle \preceq \langle\{c\#X\}, (a\,c)\cdot X\rangle$ with the substitution $\sigma = \{X \mapsto (a\,b)(a\,c)\cdot X\}$. Hence, we get $\langle\{b\#X\}, (a\,b)\cdot X\rangle \simeq \langle\{c\#X\}, (a\,c)\cdot X\rangle$, because the $\succeq$ part can be shown with the help of the substitution $\{X \mapsto (a\,c)(a\,b)\cdot X\}$.

A term-in-context $\langle \Gamma, r\rangle$ is called a *generalization* of two terms-in-context $\langle \nabla_1, t\rangle$ and $\langle \nabla_2, s\rangle$ if $\langle \Gamma, r\rangle \preceq \langle \nabla_1, t\rangle$ and $\langle \Gamma, r\rangle \preceq \langle \nabla_2, s\rangle$. It is the *least general generalization,* (lgg in short) of $\langle \nabla_1, t\rangle$ and $\langle \nabla_2, s\rangle$ if there is no generalization $\langle \Gamma', r'\rangle$ of $\langle \nabla_1, t\rangle$ and $\langle \nabla_2, s\rangle$ which satisfies $\langle \Gamma, r\rangle \prec \langle \Gamma', r'\rangle$.

Note that if we have infinite number of atoms in the language, the relation $\prec$ is not well-founded: $\langle\emptyset, X\rangle \prec \langle\{a\#X\}, X\rangle \prec \langle\{a\#X, b\#X\}, X\rangle \prec \cdots$. As a consequence, two terms-in-context may not have an lgg and not even a minimal complete set of generalizations:[1]

**Example 2.** Let $p_1 = \langle\emptyset, a_1\rangle$ and $p_2 = \langle\emptyset, a_2\rangle$ be two terms-in-context. Then in any complete set of generalizations of $p_1$ and $p_2$ there is an infinite chain $\langle\emptyset, X\rangle \prec \langle\{a_3\#X\}, X\rangle \prec \langle\{a_3\#X, a_4\#X\}, X\rangle \prec \cdots$, where $\{a_1, a_2, a_3, \ldots\}$ is the set of all atoms of the language. Hence, $p_1$ and $p_2$ do not have a minimal complete set of generalizations.

This example is a proof of the theorem, which characterizes the generalization type of nominal anti-unification:[2]

**Theorem 2.** *The problem of anti-unification for terms-in-context is of nullary type.*

However, if we restrict the set of atoms which can be used in the generalizations to be finite, then the anti-unification problem becomes unitary. (We do not prove this property here, it will follow from the Theorems 5 and 6 in Sect. VI.)

We say that a term $t$ (resp., a freshness context $\nabla$) is *based* on a set of atoms $A$ iff $\mathrm{Atoms}(t) \subseteq A$ (resp., $\mathrm{Atoms}(\nabla) \subseteq A$). A term-in-context $\langle \nabla, t\rangle$ is based on $A$ if both $t$ and $\nabla$ are based on it. We extend the notion of $A$-basedness to permutations, calling $\pi$ $A$-based if it contains only atoms from $A$. Such a permutation defines a bijection, in particular, from $A$ to $A$. If $p_1$ and $p_2$ are $A$-based terms-in-context, then their *$A$-based generalizations* are terms-in-context which are generalizations of $p_1$ and $p_2$ and are based on $A$. An *$A$-based lgg* of $A$-based terms-in-context $p_1$ and $p_2$ is a term-in-context $p$, which is an $A$-based generalization of $p_1$ and $p_2$ and there is no $A$-based generalization $p'$ of $p_1$ and $p_2$ which satisfies $p \prec p'$.

The problem we would like to solve is the following:

**Given:** Two nominal terms $t$ and $s$ of the same sort, a freshness context $\nabla$, and a *finite* set of atoms $A$ such that $t$, $s$, and $\nabla$ are based on $A$.

**Find:** A term $r$ and a freshness context $\Gamma$, such that the term-in-context $\langle \Gamma, r\rangle$ is an $A$-based least general generalization of the terms-in-context $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$.

Our anti-unification problem is parametric on the set of atoms we consider as the base, and finiteness of this set is essential to ensure the existence of an lgg.

## III. Motivation of Using a Direct Nominal Anti-Unification Algorithm

In [15], relation between nominal unification (NU) and higher-order pattern unification (HOPU) has been studied. In particular, it was shown how to translate NU problems into

[1]Minimal complete sets of generalizations are defined in the standard way. For a precise definition, see, e.g., [1], [14].

[2]Generalization types are defined analogously to unification types, see [14].

HOPU problems and how to obtain nominal unifiers back from higher-order pattern unifiers. It is tempting to use the same translation for nominal anti-unification (NAU), using the algorithm from [2] to solve higher-order anti-unification problems over patterns (HOPAU), but it turns out that the generalization computed in this way is not always based on the same set of the atoms as the input:

**Example 3.** We consider the following problem: Let the set of atoms be $A_1 = \{a, b\}$. The terms to be generalized are $a.b$ and $b.a$, and the freshness context is $\nabla = \emptyset$. According to [15], translation to higher-order patterns gives the anti-unification problem $\lambda a, b, a.\, b \triangleq \lambda a, b, b.\, a$, whose lgg is $\lambda a, b, c.X(a, b)$. However, we can not translate this lgg back to an $A_1$-based term-in-context, because it contains more bound variables than there are atoms in $A_1$.

On the other hand, the translation would work for the set of atoms $A_2 = \{a, b, c\}$: Back-translating $\lambda a, b, c.X(a, b)$ gives the $A_2$-based lgg $\langle\{c\#X\}, c.X\rangle$.

The reason why the translation-based approach does not work for $A$-based NAU is that $A$ is finite, while in higher-order anti-unification there is an infinite supply of fresh (bound) variables. If we assumed $A$ to be infinite, there would still be a mismatch between NAU and the corresponding HOPAU: NAU, as we saw, is nullary in this case, while HOPAU is unitary. The reason of this contrast is that from infinitely many nominal generalizations, there is only one which is a well-typed higher-order generalization.

One might think that the translation-based approach would still work, if one considers only nominal anti-unification problems where the set of atoms is large enough for the input terms-in-context. We have not investigated such cases, because there is another reason that speaks against NAU-to-HOPAU translation: complexity. The translation approach leads to a quadratic increase of the input size (Lemma 5.6 in [15]). The HOPAU algorithm in [2] runs in cubic time and occupies linear space with respect to the size of its input. Hence, the translation-based approach leads to an algorithm with runtime complexity $O(n^6)$ and space complexity $O(n^2)$. In contrast, the algorithm developed in this paper has runtime complexity $O(n^4)$, space complexity $O(n^2)$, and requires no back and forth translations.

## IV. The Lattice of More General Terms-In-Context

The notion of *more general term* defines an order relation between classes of terms (modulo some notion of *variable renaming*). In most cases, we have actually a meet-semilattice, since, given two terms, there always exists a greatest lower bound (meet) that corresponds to their anti-unifier. On the contrary, the least upper bound (join) of two terms only exists if they are unifiable. For instance, the two first-order terms $f(a, X_1)$ and $f(X_2, b)$ have a meet $f(Y_1, Y_2)$, and, since they are unifiable, also a join $f(a, b)$. Notice that unifiability and existence of a join are equivalent if both terms do not share variables (for instance $f(a, X)$ and $f(X, b)$ are both smaller than $f(a, b)$, hence joinable, but they are not unifiable). This restriction does not imply a loose of generality: we can reduce the unification problem $t_1 \approx^? t_2$ (sharing variables), to $f(t_1, t_2) \approx^? f(X, X)$ (not sharing variables), where $f$ is some binary symbol and $X$ a fresh variable. Therefore, in the first-order case, the problem of searching a most general unifier is equivalent to the search of the join of two terms, and the search of a least general anti-unifier to the search of the meet. Notice that meet and join are unique up to some notion of *variable renaming*. For instance, the join of $f(a, X, X')$ and $f(Y, b, Y')$ is $f(a, b, Z)$ for any renaming of $Z$ by any variable.

In the nominal case, we consider the set of terms-in-context (modulo variable renaming) with the more general relation. The following lemma establishes a correspondence between joinability and unifiability.

**Lemma 2.** *Given two terms-in-context $\langle\nabla_1, t_1\rangle$ and $\langle\nabla_2, t_2\rangle$ with disjoint sets of variables, $\langle\nabla_1, t_1\rangle$ and $\langle\nabla_2, t_2\rangle$ are joinable if, and only if, $\{t_1 \approx^? t_2\} \cup \nabla_1 \cup \nabla_2$ has a solution (is unifiable).*

*Proof:* If they are joinable, there exist $\langle\Gamma, s\rangle$ such that $\langle\nabla_i, t_i\rangle \preceq \langle\Gamma, s\rangle$. Hence, there exist $\sigma_1$ and $\sigma_2$ such that $\sigma_i$ respects $\nabla_i$ and $\nabla_i\sigma_i \subseteq \Gamma$ and $\Gamma \vdash t_i\sigma_i \approx s$. Now, since $\mathrm{Vars}(t_1) \cup \mathrm{Vars}(t_2) = \emptyset$, define $\sigma(X) = \sigma_1(X)$, if $X \in \mathrm{Vars}(t_1)$, and $\sigma(X) = \sigma_2(X)$, if $X \in \mathrm{Vars}(t_2)$. We have $\Gamma \vdash t_1\sigma \approx s \approx t_2\sigma$ and $(\nabla_1 \cup \nabla_2)\sigma \subseteq \Gamma$. Hence, according to the definition of FC, we have $\Gamma \vdash a\#X\sigma$, for any $a\#X \in \nabla_1 \cup \nabla_2$. Therefore, the pair $\langle\Gamma, \sigma\rangle$ is a nominal unifier of $\{t_1 \approx^? t_2\} \cup \nabla_1 \cup \nabla_2$.

Conversely, we can prove easily that it $\langle\Gamma, \sigma\rangle$ solves $\{t_1 \approx^? t_2\} \cup \nabla_1 \cup \nabla_2$, then $\langle\nabla_i, t_i\rangle \preceq \langle\Gamma, s\rangle$, where $s = t_1\sigma$. Let's prove now that when $\langle\Gamma, \sigma\rangle$ is the most general nominal unifier, then $\langle\Gamma, s\rangle$ is the join of $\langle\nabla_1, t_1\rangle$ and $\langle\nabla_2, t_2\rangle$. I.e. whenever $\langle\nabla_i, t_i\rangle \preceq \langle\Gamma', s'\rangle$, we have $\langle\Gamma, s\rangle \preceq \langle\Gamma', s'\rangle$:

From $\langle\nabla_i, t_i\rangle \preceq \langle\Gamma', s'\rangle$ we have that exists $\sigma'$ such that $\langle\Gamma', \sigma'\rangle$ is a nominal unifier of $\{t_1 \approx^? t_2\} \cup \nabla_1 \cup \nabla_2$ and $\Gamma' \vdash t_1\sigma' \approx s' \approx t_2\sigma'$. Since $\langle\Gamma, \sigma\rangle$ is most general, there exists a substitution $\varphi$ such that $\Gamma' \vdash \Gamma\varphi$ and $\Gamma' \vdash \sigma' \approx \varphi \circ \sigma$. The existence of this substitution $\varphi$ proves $\langle\Gamma, s\rangle \preceq \langle\Gamma', s'\rangle$. ∎

Like in first-order unification, the previous lemma allows us to reduce any nominal unification problem $P = \{a_1\#u_1, \ldots, a_m\#u_m, t_1 \approx s_1, \ldots, t_n \approx s_n\}$ into the joinability of the two terms-in-context $\langle\emptyset, f(X, X)\rangle$ and $\langle\mathsf{FC}(\{a_1\#u_1, \ldots, a_m\#u_m\}), f(g(t_1, \ldots, t_n), g(s_1, \ldots, s_n))\rangle$ where $f$ and $g$ are any appropriate function symbols, and $X$ is a fresh variable.

The nominal anti-unification problem is already stated in terms of finding the meet of two terms-in-context, with the only proviso that all terms and contexts must be based on some finite set of atoms.

## V. Nominal Anti-Unification Algorithm

The triple $X : t \triangleq s$, where $X, t, s$ have the same sort, is called the *anti-unification equation*, shortly AUE, and the

variable $X$ is called a *generalization variable*. We say that a set of AUEs $P$ is based on a finite set of atoms $A$, if for all $X : t \triangleq s \in P$, the terms $t$ and $s$ are $A$-based.

The nominal anti-unification algorithm is formulated in a rule-based way working on tuples $P;\ S;\ \Gamma;\ \sigma$ and two global parameters $A$ and $\nabla$, where

- $P$ and $S$ are sets of AUEs such that if $X : t \triangleq s \in P \cup S$, then this is the sole occurrence of $X$ in $P \cup S$;
- $P$ is the set of AUEs to be solved;
- $A$ is a finite set of atoms;
- The freshness context $\nabla$ does not constrain generalization variables;
- $S$ is a set of already solved AUEs (the store);
- $\Gamma$ is a freshness context (computed so far) which constrains generalization variables;
- $\sigma$ is a substitution (computed so far) mapping generalization variables to nominal terms;
- $P$, $S$, $\nabla$, and $\Gamma$ are $A$-based.

We call such a tuple a *system*. The rules below operate on systems.

Dec: **Decomposition**

$$\{X : \mathsf{h}(t_1, \ldots, t_m) \triangleq \mathsf{h}(s_1, \ldots, s_m)\} \cup P;\ S;\ \Gamma;\ \sigma \Longrightarrow \{Y_1 : t_1 \triangleq s_1, \ldots, Y_m : t_m \triangleq s_m\} \cup P;\ S;\ \Gamma;\ \sigma\{X \mapsto \mathsf{h}(Y_1, \ldots, Y_m)\},$$

where $\mathsf{h}$ is a function symbol or an atom, $Y_1, \ldots, Y_m$ are fresh variables of the corresponding sorts, $m \geq 0$.

Abs: **Abstraction**

$$\{X : a.t \triangleq b.s\} \cup P;\ S;\ \Gamma;\ \sigma \Longrightarrow \{Y : (c\ a)\cdot t \triangleq (c\ b)\cdot s\} \cup P;\ S;\ \Gamma;\ \sigma\{X \mapsto c.Y\},$$

where $Y$ is fresh, $c \in A$, $\nabla \vdash c\#a.t$ and $\nabla \vdash c\#b.s$.

Sol: **Solving**

$$\{X : t \triangleq s\} \cup P;\ S;\ \Gamma;\ \sigma \Longrightarrow P;\ S \cup \{X : t \triangleq s\};\ \Gamma \cup \Gamma';\ \sigma,$$

if none of the previous rules is applicable, i.e. one of the following conditions hold:

(a) both terms have distinct heads: $\mathrm{Head}(t) \neq \mathrm{Head}(s)$, or
(b) both terms are suspensions: $t = \pi_1 \cdot Y_1$ and $s = \pi_2 \cdot Y_2$, where $\pi_1, \pi_2$ and $Y_1, Y_2$ are not necessarily distinct, or
(c) both are abstractions and rule Abs is not applicable: $t = a.t'$, $s = b.s'$ and there is no atom $c \in A$ satisfying $\nabla \vdash c\#a.t'$ and $\nabla \vdash c\#b.s'$.

The set $\Gamma'$ is defined as

$$\Gamma' := \{a\#X \mid a \in A \ \wedge\ \nabla \vdash a\#t \ \wedge\ \nabla \vdash a\#s\}$$

Mer: **Merging**

$$P;\ \{X : t_1 \triangleq s_1, Y : t_2 \triangleq s_2\} \cup S;\ \Gamma;\ \sigma \Longrightarrow P;\ \{X : t_1 \triangleq s_1\} \cup S;\ \Gamma\{Y \mapsto \pi\cdot X\};\ \sigma\{Y \mapsto \pi\cdot X\},$$

where $\pi$ is an $A$-based permutation such that $\nabla \vdash \pi\cdot t_1 \approx t_2$, and $\nabla \vdash \pi\cdot s_1 \approx s_2$.

The rules transform systems to systems. One can easily observe this by inspecting the rules.

Given a finite set of atoms $A$, two nominal $A$-based terms $t$ and $s$, and an $A$-based freshness context $\nabla$, to compute $A$-based generalizations for $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, we start with $\{X : t \triangleq s\}; \emptyset; \emptyset; \varepsilon$, where $X$ is a fresh variable, and apply the rules as long as possible. We denote this procedure by $\mathfrak{N}$. A *Derivation* is a sequence of system transformations by the rules. The system to which no rule applies has the form $\emptyset; S; \Gamma; \varphi$, where Mer does not apply to $S$. We call it the *final system.* When $\mathfrak{N}$ transforms $\{X : t \triangleq s\}; \emptyset; \emptyset; \varepsilon$ into a final system $\emptyset; S; \Gamma; \varphi$, we say that the *result computed* by $\mathfrak{N}$ is $\langle \Gamma, X\varphi\rangle$.

Note that the Dec rule works also for the AUEs of the form $X : a \triangleq a$. In the Abs rule, it is important to have the corresponding $c$ in $A$. If we take $A = A_2$ in Example 3, then Abs can transform the AUE between $t$ and $s$ there, but if $A = A_1$ in the same example, then Abs is not applicable. In this case the Sol rule takes over, because the condition (c) of this rule is satisfied.

The condition (b) of Sol helps to compute, e,g, $\langle \emptyset, X\rangle$ for identical terms-in-context $\langle \emptyset, (a\ b)\cdot Y\rangle$ and $\langle \emptyset, (a\ b)\cdot Y\rangle$. Although one might expect that computing $\langle \emptyset, (a\ b)\cdot Y\rangle$ would be more natural, from the generalization point of view it does not matter, because $\langle \emptyset, X\rangle$ is as general as $\langle \emptyset, (a\ b)\cdot Y\rangle$.

We illustrate $\mathfrak{N}$ with the help of a couple of more examples:

**Example 4.** Let $t = f(a, b)$, $s = f(b, c)$, $\nabla = \emptyset$, and $A = \{a, b, c, d\}$. Then $\mathfrak{N}$ performs the following transformations:

$$\{X : f(a,b) \triangleq f(b,c)\}; \emptyset; \emptyset; \varepsilon \Longrightarrow_{\mathsf{Dec}}$$
$$\{Y : a \triangleq b, Z : b \triangleq c\}; \emptyset; \emptyset; \{X \mapsto f(Y,Z)\} \Longrightarrow^{2}_{\mathsf{Sol}}$$
$$\emptyset; \{Y : a \triangleq b, Z : b \triangleq c\}; \{c\#Y, d\#Y, a\#Z, d\#Z\};\ \{X \mapsto f(Y,Z)\} \Longrightarrow_{\mathsf{Mer}}$$
$$\emptyset; \{Y : a \triangleq b\}; \{c\#Y, d\#Y\}; \{X \mapsto f(Y, (a\,b)(b\,c)\cdot Y)\}$$

Hence, $p = \langle \{c\#Y, d\#Y\}, f(Y, (a\,b)(b\,c)\cdot Y)\rangle$ is the computed result. It generalizes the input pairs: $p\{Y \mapsto a\} \preceq \langle \nabla, t\rangle$ and $p\{Y \mapsto b\} \preceq \langle \nabla, s\rangle$. The substitutions $\{Y \mapsto a\}$ and $\{Y \mapsto b\}$ can be read from the final store. Note that $\langle \{c\#Y\}, f(Y, (a\,b)(b\,c)\cdot Y)\rangle$ would be also an $A$-based generalization of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, but it is strictly more general than $p$.

**Example 5.** Let $t = f(b, a)$, $s = f(Y, (a\,b)\cdot Y)$, $\nabla = \{b\#Y\}$, and $A = \{a, b\}$. Then $\mathfrak{N}$ computes the term-in-context $p = \langle \emptyset, f(Z_1, (a\,b)\cdot Z_1)\rangle$. It generalizes the input pairs.

**Example 6.** Let $t = f(a.b, X)$, $s = f(b.a, Y)$, $\nabla = \{c\#X\}$, $A = \{a, b, c, d\}$. Then $\mathfrak{N}$ computes the term-in-context $p = \langle \{c\#Z_1, d\#Z_1\}, f(c.Z_1, Z_2)\rangle$. It generalizes the input pairs: $p\{Z_1 \mapsto b, Z_2 \mapsto X\} = \langle \emptyset, f(c.b, X)\rangle \preceq \langle \nabla, t\rangle$ and $p\{Z_1 \mapsto a, Z_2 \mapsto Y\} = \langle \emptyset, f(c.a, Y)\rangle \preceq \langle \nabla, s\rangle$.

## VI. Properties of the Nominal Anti-Unification Algorithm

**Theorem 3** (Termination of $\mathfrak{N}$)**.** *The procedure $\mathfrak{N}$ terminates on any input (provided that the computation of $\pi$ in the Merge*

*rule terminates).*

*Proof:* We associate to each system $P;\ S;\ \nabla;\ A;\ \Gamma;\ \sigma$ its measure, a triple $\langle n, M(P), M(S)\rangle$, where $n$ is a number of abstractions in $P$, and $M(U)$ is a multiset defined for a set of AUEs $U$ as follows:

$$M(U) := \{\|s\| + \|t\| \mid X : t \triangleq s \in U \text{ for some } X\}.$$

Measures are compared lexicographically. Obviously, each rule in $\mathfrak{N}$ strictly reduces it. The ordering is well-founded. In the conditions of the rules, proving atomic freshness formulas from freshness contexts terminates. Computation of $\pi$ in the Merge rule terminates by assumption. Hence, $\mathfrak{N}$ terminates. ∎

In the next section we design a procedure which computes the permutations needed in the condition of the Mer rule and prove its termination. It will fulfill the proviso of Theorem 3.

The Soundness Theorem states that the result computed by $\mathfrak{N}$ is indeed an $A$-based generalization of the input terms-in-context:

**Theorem 4** (Soundness of $\mathfrak{N}$)**.** *Given terms $t$ and $s$ and a freshness context $\nabla$, all based on a finite set of atoms $A$, if $\{X : t \triangleq s\};\ \emptyset;\ \emptyset;\ \varepsilon \Longrightarrow^{+} \emptyset;\ S;\ \Gamma;\ \sigma$ is a derivation obtained by an execution of $\mathfrak{N}$, then $\langle \Gamma, X\sigma\rangle$ is an $A$-based generalization of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$.*

*Proof:* See the appendix. ∎

The Completeness Theorem states that for any $A$-based generalization of the input terms-in-context, $\mathfrak{N}$ can compute one which is at most as general that the given generalization.

**Theorem 5** (Completeness of $\mathfrak{N}$)**.** *Given terms $t$ and $s$ and freshness contexts $\nabla$ and $\Gamma$, all based on a finite set of atoms $A$, if $\langle \Gamma, r\rangle$ is an $A$-based generalization of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, then there exists a derivation $\{X : t \triangleq s\};\ \emptyset;\ \emptyset;\ \varepsilon \Longrightarrow^{+} \emptyset;\ S;\ \Gamma';\ \sigma$ obtained by an execution of $\mathfrak{N}$, such that $\langle \Gamma, r\rangle \preceq \langle \Gamma', X\sigma\rangle$.*

*Proof:* See the appendix. ∎

Depending on the selection of AUEs to perform a step, there can be different derivations in $\mathfrak{N}$ starting from the same AUE, leading to different generalizations. The next theorem states that all those generalizations are the same modulo variable renaming and $\alpha$-equivalence.

**Theorem 6** (Uniqueness Modulo $\simeq$)**.** *Let $t$ and $s$ be terms and $\nabla$ be a freshness context that are based on the same finite set of atoms. Let $\{X : t \triangleq s\};\ \emptyset;\ \emptyset;\ \varepsilon \Longrightarrow^{+} \emptyset;\ S_1;\ \Gamma_1;\ \sigma_1$ and $\{X : t \triangleq s\};\ \emptyset;\ \emptyset;\ \varepsilon \Longrightarrow^{+} \emptyset;\ S_2;\ \Gamma_2;\ \sigma_2$ be two maximal derivations in $\mathfrak{N}$. Then $\langle \Gamma_1, X\sigma_1\rangle \simeq \langle \Gamma_2, X\sigma_2\rangle$.*

*Proof:* See the appendix. ∎

Theorems 4, 5, and 6 imply that nominal anti-unification is unitary: For any $A$-based $\nabla$, $t$, and $s$, there exists an $A$-based lgg of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, which is unique modulo $\simeq$ and can be computed by the algorithm $\mathfrak{N}$.

Now we study how lgg's of terms-in-context depend on the set of atoms the terms-in-context are based on. The following lemma states the precise dependence.

**Lemma 3.** *Let $A_1$ and $A_2$ be two finite sets of atoms with $A_1 \subseteq A_2$ such that the $A_1$-based terms-in-context $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$ have an $A_1$-based lgg $\langle \Gamma_1, r_1\rangle$ and an $A_2$-based lgg $\langle \Gamma_2, r_2\rangle$. Then $\Gamma_2 \vdash r_1 \preceq r_2$.*

*Proof:* $\langle \Gamma_1, r_1\rangle$ and $\langle \Gamma_2, r_2\rangle$ are unique modulo $\simeq$. Let $\mathsf{D}_i$ be the derivation in $\mathfrak{N}$ that computes $\langle \Gamma_i, r_i\rangle$, $i = 1, 2$. The number of atoms in $A_1$ and $A_2$ makes a difference in the rule Abs: If there are not enough atoms in $A_1$, an Abs step in $\mathsf{D}_2$ is replaced by a Sol step in $\mathsf{D}_1$. It means that for all positions $\mathsf{p}$ of $r_1$, $r_2|_{\mathsf{p}}$ is also defined. Moreover, there might exist a subterm $r_1|_{\mathsf{p}}$, which has a form of suspension, while $r_2|_{\mathsf{p}}$ is an abstraction. For such positions, $r_1|_{\mathsf{p}} \preceq r_2|_{\mathsf{p}}$. For the other positions $\mathsf{p}'$ of $r_1$, $r_1|_{\mathsf{p}'}$ and $r_2|_{\mathsf{p}'}$ may differ only by names of generalization variables or by names of bound atoms.

Another difference might be in the application of Sol in both derivations: It can happen that this rule produces a larger $\Gamma'$ in $\mathsf{D}_2$ than in $\mathsf{D}_1$, when transforming the same AUE.

Hence, if there are positions $\mathsf{p}_1, \ldots, \mathsf{p}_n$ in $r_1$ such that $r_1|_{\mathsf{p}_i} = \pi_i \cdot X$, then there exists a substitution $\varphi_X$ such that $\Gamma_2 \vdash \pi_i \cdot X\varphi \approx r_2|_{\mathsf{p}_i}$, $1 \le i \le n$. Taking the union of all $\varphi_X$'s where $X \in \mathrm{Vars}(r_1)$, we get $\varphi$ with the property $\Gamma_2 \vdash r_1\varphi \approx r_2$. ∎

Note that, in general, we can not replace $\Gamma_2 \vdash r_1 \preceq r_2$ with $\Gamma_2 \vdash r_1 \simeq r_2$ in Lemma 3. The following example illustrates this:

**Example 7.** Let $t = a.b$, $s = b.a$, $\nabla = \emptyset$, $A_1 = \{a, b\}$, and $A_2 = \{a, b, c\}$. Then for $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, $\langle \emptyset, X\rangle$ is an $A_1$-based lgg and $\langle \{c\#X\}, c.X\rangle$ is an $A_2$-based lgg. Obviously, $\{c\#X\} \vdash X \preceq c.X$ but not $\{c\#X\} \vdash c.X \preceq X$.

This example naturally leads to a question: Under which additional conditions can we have $\Gamma_2 \vdash r_1 \simeq r_2$ instead of $\Gamma_2 \vdash r_1 \preceq r_2$ in Lemma 3? To formalize a possible answer to it, we need some notation.

Let the terms $t, s$ and the freshness context $\nabla$ be based on the same set of atoms $A$. The maximal subset of $A$, *fresh* for $t, s$, and $\nabla$, denoted $\mathit{fresh}(A, t, s, \nabla)$, is defined as $A \setminus (\mathrm{Atoms}(t) \cup \mathrm{Atoms}(s) \cup \mathrm{Atoms}(\nabla))$.

If $A_1 \subseteq A_2$ are two sets of atoms such that $t, s, \nabla$ are at the same time based on both $A_1$ and $A_2$, then $\mathit{fresh}(A_1, t, s, \nabla) \subseteq \mathit{fresh}(A_2, t, s, \nabla)$.

Let $\|t\|_{\mathsf{Abs}}$ stand for the number of abstraction occurrences in $t$. $|A|$ stands for the cardinality of the set of atoms $A$. We say that a set of atoms $A$ is *saturated* for $A$-based $t, s$ and $\nabla$, if $|\mathit{fresh}(A, t, s, \nabla)| \ge \min\{\|t\|_{\mathsf{Abs}}, \|s\|_{\mathsf{Abs}}\}$.

The following lemma answers the question posed above:

**Lemma 4.** *Under the conditions of Lemma 3, if $A_1$ is saturated for $t, s, \nabla$, then $\Gamma_2 \vdash r_1 \simeq r_2$.*

*Proof:* Let $\mathsf{D}_i$ be the derivation in $\mathfrak{N}$ that computes $\langle \Gamma_i, r_i\rangle$, $i = 1, 2$. Note that in each of these derivations, the number of Abs steps does not exceed $\min\{\|t\|_{\mathsf{Abs}}, \|s\|_{\mathsf{Abs}}\}$. Since $A_1$ is saturated for $t, s, \nabla$ and $A_1 \subseteq A_2$, $A_2$ is also saturated for $t, s, \nabla$. Hence, whenever an AUE between two abstractions is encountered in the derivation $\mathsf{D}_i$, there is always

$c \in A_1$ available which satisfies the condition of the Abs rule. Therefore, such AU-E's are never transformed by Sol. We can assume without loss of generality that the sequence of steps in $\mathsf{D}_1$ and $\mathsf{D}_2$ are the same. we may also assume that we take the same fresh variables, and the same atoms from $\mathit{fresh}(A_1, t, s, \nabla)$ in the corresponding steps in $\mathsf{D}_1$ and $\mathsf{D}_2$. Then the only difference between these derivations is in the $\Gamma$'s, caused by the Sol rule which might eventually make $\Gamma_2$ larger than $\Gamma_1$. The $\sigma$'s computed by the derivations are the same and, therefore, $r_1$ and $r_2$ are the same (modulo the assumptions on the variable and fresh atom names). Hence, $\Gamma_2 \vdash r_1 \simeq r_2$. ∎

## VII. Deciding Equivariance

Computation of $\pi$ in the condition of the rule Mer above requires an algorithm that solves the following problem: Given a finite set of atoms $A$, terms $t$ and $s$, and a freshness context $\nabla$, all based on $A$, find an $A$-based permutation $\pi$ such that $\nabla \vdash \pi{\cdot}t \approx s$. This is the problem of deciding whether $t$ and $s$ are equivariant with respect to $\nabla$ and $A$. In this section we describe an algorithm that solves this problem by effectively computing the corresponding permutation.

This rule-base algorithm, which we call $\mathfrak{E}$, works on tuples of the form $E;\ \nabla;\ A;\ \pi$ (also called systems). $E$ is a set of equivariance equations of the form $t \approx s$ where $t, s$ are nominal terms, $\nabla$ is a freshness context, and $A$ is a finite set of atoms which are available for computing $\pi$. The latter holds the permutation to be returned in case of success.

The algorithm is split into two phases. The first one is a simplification phase where function applications, abstractions and suspensions are decomposed as long as possible. The second phase is the permutation computation, where given a set of equivariance equations between atoms of the form $a \approx b$ we compute the permutation which will be returned in case of success.

The rules of the first phase are the following:

Dec-E: **Decomposition**

$$\{f(t_1, \ldots, t_m) \approx f(s_1, \ldots, s_m)\} \uplus E;\ \nabla;\ A;\ \mathit{Id} \Longrightarrow \{t_1 \approx s_1, \ldots, t_m \approx s_m\} \cup E;\ \nabla;\ A;\ \mathit{Id}.$$

Alp-E: **Alpha Equivalence**

$$\{a.t \approx b.s\} \uplus E;\ \nabla;\ A;\ \mathit{Id} \Longrightarrow \{(a\ \acute{c}){\cdot}t \approx (b\ \acute{c}){\cdot}s\} \cup E;\ \nabla;\ A;\ \mathit{Id},$$

where $\acute{c}$ is a fresh atom of the same sort as $a$ and $b$.

Sus-E: **Suspension**

$$\{\pi_1{\cdot}X \approx \pi_2{\cdot}X\} \uplus E;\ \nabla;\ A;\ \mathit{Id} \Longrightarrow \{\pi_1{\cdot}a \approx \pi_2{\cdot}a \mid a \in A \wedge a\#X \notin \nabla\} \cup E;\ \nabla;\ A;\ \mathit{Id}.$$

The rules of the second phase are the following:

Rem-E: **Remove**

$$\{a \approx b\} \uplus E;\ \nabla;\ A;\ \pi \Longrightarrow E;\ \nabla;\ A \setminus \{b\};\ \pi,$$

if $\pi{\cdot}a = b$.

Sol-E: **Solve**

$$\{a \approx b\} \uplus E;\ \nabla;\ A;\ \pi \Longrightarrow E;\ \nabla;\ A \setminus \{b\};\ (\pi{\cdot}a\ b)\pi,$$

if $\pi{\cdot}a, b \in A$ and $\pi{\cdot}a \neq b$.

Note that in Alp-E, $\acute{c}$ is fresh means that $\acute{c} \notin A$ and, therefore, $\acute{c}$ will not appear in $\pi$. These atoms are an auxiliary means which play a role during the computation but do not appear in the final result. Implicitly, we assume that $\acute{c}\#X \in \nabla$ for any $\acute{c} \notin A$ and any $X \in \mathrm{Vars}(t, s)$.

The input for $\mathfrak{E}$ is initialized in the Mer rule, which needs to compute an $A$-based permutation $\pi$ for $A$-based context $\nabla$ and two AUEs $X : t_1 \triangleq s_1$ and $Y : t_2 \triangleq s_2$. We create the initial system $\{t_1 \approx t_2, s_1 \approx s_2\}$;[3] $\nabla;\ A;\ \mathit{Id}$ and in the first phase apply exhaustively the rules Dec-E, Alp-E and Sus-E until (eventually) we get a set of equivariance equations between atoms. Then, we apply Rem-E and Sol-E as long as possible. This application will not give rise to a problem to which Dec-E, Alp-E, or Sus-E applies. If the final system is the *success state* $\emptyset;\ \nabla;\ A;\ \pi$, then we say that the $\mathfrak{E}$ *computes* the permutation $\pi$. Otherwise the obtained system has the form $E;\ \nabla;\ A;\ \pi$ with $E \neq \emptyset$ to which no rule applies. It is transformed into $\bot$, called the *failure state*.

We illustrate the algorithm $\mathfrak{E}$ on examples:

**Example 8.** Consider the equivariance problem $E = \{a \approx a,\ a.(a\,b)(c\,d){\cdot}X \approx b.X\}$, $A = \{a, b, c, d\}$, and $\nabla = \{a\#X\}$:

$$\{a \approx a, a.(a\,b)(c\,d){\cdot}X \approx b.X\};\ \{a\#X\};\ \{a, b, c, d\};\ \mathit{Id} \Longrightarrow_{\text{Alp-E}}$$
$$\{a \approx a, (a\,\acute{e})(a\,b)(c\,d){\cdot}X \approx (b\,\acute{e}){\cdot}X\};\ \{a\#X\};\ \{a, b, c, d\};\ \mathit{Id} \Longrightarrow_{\text{Sus-E}}$$
$$\{a \approx a, \acute{e} \approx \acute{e}, c \approx d, d \approx c\};\ \{a\#X\};\ \{a, b, c, d\};\ \mathit{Id} \Longrightarrow_{\text{Rem-E}}$$
$$\{\acute{e} \approx \acute{e}, c \approx d, d \approx c\};\ \{a\#X\};\ \{b, c, d\};\ \mathit{Id} \Longrightarrow_{\text{Rem-E}}$$
$$\{c \approx d, d \approx c\};\ \{a\#X\};\ \{b, c, d\};\ \mathit{Id} \Longrightarrow_{\text{Sol-E}}$$
$$\{d \approx c\};\ \{a\#X\};\ \{b, c\};\ (c\,d) \Longrightarrow_{\text{Rem-E}}$$
$$\emptyset;\ \{a\#X\};\ \{b\};\ (c\,d).$$

**Example 9.**

- For $E = \{a.f(b, X) \approx b.f(a, X)\}$; $A = \{a, b\}$, and $\nabla = \{a\#X\}$, $\mathfrak{E}$ returns $\bot$.
- For $E = \{a.f(b, (a\,b){\cdot}X) \approx b.f(a, X)\}$, $A = \{a, b\}$, and $\nabla = \{a\#X\}$, $\mathfrak{E}$ returns $(b\,a)$.
- For $E = \{a.b.(a\,b)(a\,c){\cdot}X = b.a.(a\,c){\cdot}X\}$, $A = \{a, b\}$, and $\nabla = \emptyset$, $\mathfrak{E}$ returns $\mathit{Id}$.
- For $E = \{a.b.(a\,b)(a\,c){\cdot}X = a.b.(b\,c){\cdot}X\}$, $A = \{a, b\}$, and $\nabla = \emptyset$, $\mathfrak{E}$ returns $\bot$.

**Theorem 7** (Termination of $\mathfrak{E}$)**.** *The procedure $\mathfrak{E}$ terminates on any input.*

*Proof:* We define the complexity measure of a quadruple $E;\ \nabla;\ A;\ \pi$ as a tuple of multisets $(M_1(E), M_2(E))$, where

$$M_1(E) ::= \{\|s\|_{\mathrm{Vars}} + \|t\|_{\mathrm{Vars}} \mid s \approx t \in E\},$$

[3] Equivalently, we could also initialize the set of equivariance equations with $\{f(t_1, t_2) \approx f(s_1, s_2)\}$ for some $f$.

$$M_2(E) ::= \{\|s\| + \|t\| \mid s \approx t \in E\}.$$

The measures are compared by the well-founded lexicographic ordering. Each rule strictly reduces the complexity. ∎

The Soundness Theorem for $\mathfrak{E}$ states that the permutation the algorithm computes, indeed, shows that the input terms are equivariant:

**Theorem 8** (Soundness of $\mathfrak{E}$)**.** *Let* $\{t \approx s\}; \nabla; A; Id \Longrightarrow^* \emptyset; \nabla; B; \pi$ *be a derivation in* $\mathfrak{E}$*, then* $\pi$ *is an* $A$*-based permutation such that* $\nabla \vdash \pi \cdot t \approx s$*.*

*Proof:* See the appendix. ∎

We now prove a lemma which asserts an invariant of single-step transformations with the rules of $\mathfrak{E}$:

**Lemma 5** (First Invariant Lemma)**.** *Let* $A$ *be a finite set of atoms,* $E_1$ *be a set of equivariance equations for terms based on* $A$*,* $\pi_1$ *be an* $A$*-based permutation and* $A_1 \subseteq A$*. Let* $\mu$ *be an* $A$*-based permutation such that* $\nabla \vdash \mu \cdot t \approx s$*, for all* $t \approx s \in E_1$*. If* $E_1; \nabla; A_1; \pi_1 \Longrightarrow E_2; \nabla; A_2; \pi_2$ *is a step performed by a rule in* $\mathfrak{E}$*, then*

1) $\nabla \vdash \mu \cdot t' \approx s'$*, for all* $t' \approx s' \in E_2$*.*
2) *If* $\mu^{-1} \cdot b = \pi_1^{-1} \cdot b$*, for all* $b \in A \setminus A_1$*, then* $\mu^{-1} \cdot b = \pi_2^{-1} \cdot b$*, for all* $b \in A \setminus A_2$*.*

*Proof:* By case distinction on the applied rule.

Dec-E: The proposition is obvious.

Alp-E: In this case it follows from the definitions of $\approx$ and permutation application.

Sus-E: In this case $t = \tau_1 \cdot X$, $s = \tau_2 \cdot X$, $\pi_1 = Id$, and by the assumption we have $\nabla \vdash \mu\tau_1 \cdot X \approx \tau_2 \cdot X$. By the definition of $\approx$, it means that the following statements hold:

(a) For all $a$, if $\tau_1 \cdot a \neq \tau_2 \cdot a$ and $a \# X \notin \nabla$, then $\mu\tau_1 \cdot a = \tau_2 \cdot a$.
(b) For all $a$, if $\tau_1 \cdot a = \tau_2 \cdot a$ and $a \# X \notin \nabla$, then $\mu\tau_1 \cdot a = \tau_1 \cdot a$.

From (a) we get that $\mu \cdot a = b$, for all pairs $a \neq b$ such that exist an atom $c$ such that $a = \tau_1 \cdot c$, $b = \tau_2 \cdot c$ and $c \# X \notin \nabla$. This implies the item 1 of the lemma.

To prove the item 2 of the lemma, first note that since $\pi_1 = Id$ when the rule Sus-E is applied, by the assumption of this item we have $\mu^{-1} \cdot b = \pi_1^{-1} \cdot b = b$, for all $b \in A \setminus A_1$. Also, $\pi_2 = Id$, therefore $\pi_1^{-1} \cdot b = b$ for all $b \in A \setminus A_2$. Thus, we need to show $\mu^{-1} \cdot b = b$ for all $b \in A \setminus A_2$. Since $A_2 \subseteq A_1 \subseteq A$, we only need to prove $\mu^{-1} \cdot b = b$, but this directly follows from (b).

Rem-E: The item 1 is trivial. To prove the item 2, note that $t = a$, $s = b$, $\pi_1 = \pi_2$ and we only need to show $\mu^{-1} \cdot b = \pi_2^{-1} \cdot b$. By the assumption we have $\nabla_1 \vdash \mu \cdot a \approx b$. Since $a$ and $b$ are atoms, the latter simply means that $\mu \cdot a = b$. From the rule condition we also know that $\pi_1 \cdot a = b$. From these two equalities we get $\mu^{-1} \cdot b = a = \pi_2^{-1} b$.

Sol-E: The item 1 is trivial also in this case. To prove the item 2, note that $t = a$, $s = b$, $\pi_2 = (\pi_1 \cdot a\ b)\pi_1$ and we only need to show $\mu^{-1} \cdot b = \pi_2^{-1} \cdot b$. By the assumption we have $\nabla \vdash \mu \cdot a \approx b$, which means that $\mu \cdot a = b$ and, hence, $a = \mu^{-1} \cdot b$. As for $\pi_2^{-1} \cdot b$, we have $\pi_2^{-1} \cdot b = \pi_1^{-1}(\pi_1 \cdot a\ b) \cdot b = \pi_1^{-1} \cdot ((\pi_1 \cdot a\ b) \cdot b) = \pi_1^{-1}\pi_1 \cdot a = a$. Hence, we get $\mu^{-1} \cdot b = a = \pi_2^{-1} \cdot b$. ∎

The following invariant is maintained by the algorithm $\mathfrak{E}$:

**Lemma 6** (Second Invariant Lemma)**.** *Let* $A$ *be a finite set of atoms,* $E_1$ *be a set of equivariance equations,* $\pi_1$ *be a permutation and* $A_1 \subseteq A$*, all of them* $A$*-based. Let* $\mu$ *be an* $A$*-based permutation such that* $\nabla \vdash \mu \cdot t \approx \cdot s$*, for all* $t \approx s \in E_1$*. If* $E_1; \nabla; A_1; \pi_1 \Longrightarrow^+ E_2; \nabla; A_2; \pi_2$ *is a sequence of steps performed by* $\mathfrak{E}$*, then*

1) $\nabla \vdash \mu \cdot t' \approx \cdot s'$*, for all* $t' \approx s' \in E_2$*.*
2) *If* $\mu^{-1} \cdot b = \pi_1^{-1} \cdot b$*, for all* $b \in A \setminus A_1$*, then* $\mu^{-1} \cdot b = \pi_2^{-1} \cdot b$*, for all* $b \in A \setminus A_2$*.*

*Proof:* By induction on the length of the sequence, using Lemma 5 and the definition of rules of $\mathfrak{E}$. ∎

These lemmas are used in the proof of the Completeness Theorem for $\mathfrak{E}$:

**Theorem 9** (Completeness of $\mathfrak{E}$)**.** *Let* $A$ *be a finite set of atoms,* $t, s$ *be* $A$*-based terms, and* $\nabla$ *be a* $A$*-based freshness context. If* $\nabla \vdash \mu \cdot t \approx s$ *holds for some* $A$*-based permutation* $\mu$*, then there exists a derivation* $\{t \approx s\}; \nabla; A; Id \Longrightarrow^* \emptyset; \Gamma; B; \pi$*, obtained by an execution of* $\mathfrak{E}$*, such that* $\pi \cdot a = \mu \cdot a$ *for any atom* $a \in \mathrm{FA}(t)$*.*

*Proof:* See the appendix. ∎

## VIII. Complexity Analysis

We represent a permutations $\pi$ as two hash tables. One for the permutation itself, we call it $T_\pi$, and one for the inverse of the permutation, called $T_{\pi^{-1}}$. The key of a hash tables is an atom and we associate another atom, the mapping, with it. For instance the permutation $\pi = (a\,b)(a\,c)$ is represented as $T_\pi = \{a \mapsto c, b \mapsto a, c \mapsto b\}$ and $T_{\pi^{-1}} = \{a \mapsto b, b \mapsto c, c \mapsto a\}$. We write $T_\pi(a)$ to obtain from the hash table $T_\pi$ the atom which is associated with the key $a$. If no atom is associated with the key $a$ then $T_\pi(a)$ returns $a$. We write $T_\pi(a \mapsto b)$, to set the mapping such that $T_\pi(a) = b$. As the set of atoms is small, we can assume a perfect hash function. It follows, that both defined operations are done in constant time, leading to constant time application of a permutation. Swapping application to a permutation $(a\,b)\pi$ is also done in constant time in the following way: Obtain $c = T_{\pi^{-1}}(a)$ and $d = T_{\pi^{-1}}(b)$ and perform the following updates:

(a) $T_\pi(c \mapsto b)$ and $T_\pi(d \mapsto a)$,
(b) $T_{\pi^{-1}}(b \mapsto c)$ and $T_{\pi^{-1}}(a \mapsto d)$.

We also represent set membership of atoms to a set of atoms $A$ with a hash table $\in_A$ from atoms to Booleans such that $\in_A (a) = true$ iff $a \in A$. We also have a list $L_A$ of the atoms representing the entries of the table such that $\in_A (a) = true$ to easily know all atoms in $A$.

Finally we also represent set membership of freshness constraints to a freshness environment $\nabla$ with a hash table $\in_\nabla$.

**Theorem 10.** *The equivariance algorithm* $\mathfrak{E}$ *has quadratic space and time complexity.*

*Proof:* Given a set of atoms $A$, an equivariance problem $E$ and a freshness constraint $\nabla$, both $A$-based, let $n = \max\{|A|, |E|, |\nabla|\}$.

We analyze complexity of both phases.

For the first phase, notice that all rules can be applied only $O(n)$ many times, since Dec-E removes two function symbols and Alp-E two abstraction, and Sus-E two suspensions. The resulting equations after this phase only contain atoms. However, notice that the size of this equations is not necessarily linear. Every time we apply Alp-E a new swapping is applied to both subterms. This swappings may increase the size of suspensions occurring bellow the abstraction. Since there are $O(n)$ many suspensions and $O(n)$ many abstractions, the final size of suspensions is $O(n^2)$. This is the size of the atom equations at the beginning of the second phase. We can see that the application of Dec-E rule has $O(1)$ time complexity (with the appropriate representation of equations).

The application of Alp-E rule requires to find a fresh atom (not necessarily from $A$, hence $O(1)$) and it has to apply a swapping twice. This application requires traversing the term hence has $O(n)$ time complexity. The application of Sus-E requires to traverse $L_A$ ($O(n)$) and check for freshness membership in $\in_\nabla$ ($O(1)$). Finally it has to add equations like $(\pi_1 \cdot a \approx \pi_2)$, this requires to build $T_{\pi_1}$ and $T_{\pi_2}$ that can be done in $O(n)$ time complexity and allow us to build each equation in $O(1)$ time.

Summing up, this phase has $O(n^2)$ time complexity and $O(n^2)$ space complexity because it can increase quadratically the number of equations due to Sus-E applications.

For the second phase, notice that both rules Rem-E and Sol-E remove an equation and do not introduce any other one. Hence, potentially having $O(n^2)$ many equations in this phase, these equations can be applied $O(n^2)$ may times. We construct a hash table $T_\pi$ for $\pi$ that will be maintained and used by both rules. Fortunately, each application has time complexity $O(1)$. Rem-E uses $T_\pi$ to check for applicability and in case it is applied only removes $b$ from $A$, hence updates $\in_A$ (notice that we don't care about $L_A$ in this second phase of the algorithm). Sol-E uses $\in_A$ and $T_\pi$ to check for applicability and in case it is applied only removes $b$ from $A$ (hence updates $\in_A$), and updates $T_\pi$.

Summing up this phase maintains the overall $O(n^2)$ time complexity and $O(n^2)$ space complexity of the algorithm. ■

**Theorem 11.** *The nominal anti-unification algorithm $\mathfrak{N}$ has $O(n^4)$ time complexity and $O(n^2)$ space complexity.*

*Proof:* Rules Dec and Abs can be applied $O(n)$ many times and strictly reduce the size of the problem. Rule Sol also reduces the size of the problem on expenses of moving equation to the store $S$. Hence, the size of the store is $O(n)$. Since rule Mer can be checked for any pair of equations in the store, we could have $O(n^2)$ applications (or trials of applications) of this rule. Since this rule calls the equivariance algorithm, and this has $O(n^2)$ time and space complexity, this results into the stated bounds. ■


## ACKNOWLEDGMENT

This research has been partially supported by the projects HeLo (TIN2012-33042) and TASSAT (TIN2010-20967-C04-01), by the Austrian Science Fund (FWF) with the project SToUT (P 24087-N18).


## REFERENCES


[1] M. Alpuente, S. Escobar, J. Meseguer, and P. Ojeda. A modular equational generalization algorithm. In M. Hanus, editor, *LOPSTR*, volume 5438 of *Lecture Notes in Computer Science*, pages 24–39. Springer, 2008.

[2] A. Baumgartner, T. Kutsia, J. Levy, and M. Villaret. A variant of higher-order anti-unification. In F. van Raamsdonk, editor, *RTA*, volume 21 of *LIPIcs*, pages 113–127. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[3] C. Calvès. *Complexity and Implementation of Nominal Algorithms*. PhD thesis, Kings College London, 2010.

[4] C. Calvès and M. Fernández. A polynomial nominal unification algorithm. *Theor. Comput. Sci.*, 403(2-3):285–306, 2008.

[5] C. Calvès and M. Fernández. Matching and alpha-equivalence check for nominal terms. *J. Comput. Syst. Sci.*, 76(5):283–301, 2010.

[6] J. Cheney. Equivariant unification. *JAR*, 45(3):267–300, 2010.

[7] J. Cheney and C. Urban. alpha-Prolog: A logic programming language with names, binding and a-equivalence. In B. Demoen and V. Lifschitz, editors, *ICLP*, volume 3132 of *Lecture Notes in Computer Science*, pages 269–283. Springer, 2004.

[8] G. Dowek, M. J. Gabbay, and D. P. Mulligan. Permissive nominal terms and their unification: an infinite, co-infinite approach to nominal techniques. *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[9] C. Feng and S. Muggleton. Towards inductive generalization in higher order logic. In D. H. Sleeman and P. Edwards, editors, *ML*, pages 154–162. Morgan Kaufmann, 1992.

[10] M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

[11] M. J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, University of Cambridge, UK, 2000.

[12] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. In *LICS*, pages 214–224. IEEE Computer Society, 1999.

[13] U. Krumnack, A. Schwering, H. Gust, and K.-U. Kühnberger. Restricted higher-order anti-unification for analogy making. In M. A. Orgun and J. Thornton, editors, *Australian Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, pages 273–282. Springer, 2007.

[14] T. Kutsia, J. Levy, and M. Villaret. Anti-unification for unranked terms and hedges. In M. Schmidt-Schauß, editor, *RTA*, volume 10 of *LIPIcs*, pages 219–234. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[15] J. Levy and M. Villaret. Nominal unification from a higher-order perspective. *ACM Trans. Comput. Log.*, 13(2):10, 2012.

[16] J. Lu, J. Mylopoulos, M. Harao, and M. Hagiya. Higher order generalization and its application in program verification. *Ann. Math. Artif. Intell.*, 28(1-4):107–126, 2000.

[17] D. Mulligan. *Extensions of Nominal Terms*. PhD thesis, School of Math. and Comp. Sci., Heriot-Watt University, Edinburgh, 2011.

[18] F. Pfenning. Unification and anti-unification in the calculus of constructions. In *LICS*, pages 74–85. IEEE Computer Society, 1991.

[19] G. D. Plotkin. A note on inductive generalization. *Machine Intel.*, 5(1):153–163, 1970.

[20] L. D. Raedt. *Logical and Relational Learning*. Springer, 2008.

[21] J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intel.*, 5(1):135–151, 1970.

[22] U. Schmid. *Inductive Synthesis of Functional Programs, Universal Planning, Folding of Finite Programs, and Schema Abstraction by Analogical Reasoning*, volume 2654 of *Lecture Notes in Computer Science*. Springer, 2003.

[23] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unificaiton. In M. Baaz and J. A. Makowsky, editors, *CSL*, volume 2803 of *Lecture Notes in Computer Science*, pages 513–527. Springer, 2003.

[24] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. *Theor. Comput. Sci.*, 323(1–3):473–497, 2004.

## APPENDIX

**Theorem 4** (Soundness of $\mathfrak{N}$)**.** *context $\nabla$, all based on a finite set of atoms $A$, if $\{X : t \triangleq s\}; \emptyset; \emptyset;\ \varepsilon \Longrightarrow^{+} \emptyset;\ S;\ \Gamma;\ \sigma$ is a derivation obtained by an execution of $\mathfrak{N}$, then $\langle \Gamma, X\sigma\rangle$ is an $A$-based generalization of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$.*

*Proof:* Since all atoms introduced by the rules of $\mathfrak{N}$ in $\Gamma$ and $\sigma$ are from $A$, $\langle \Gamma, X\sigma\rangle$ is $A$-based. To prove that $\langle \Gamma, X\sigma\rangle$ generalizes both $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, we use well-founded induction on the length of derivations. In fact, we will prove a more general statement:

Assume $P_0; S_0; \Gamma_0; \vartheta_0 \Longrightarrow^{+} \emptyset; S_n; \Gamma_n; \vartheta_0\vartheta_1\cdots\vartheta_n$ is a derivation in $\mathfrak{N}$ (with $\nabla$ and $A$) with the following property: If $Z_0 : t_0 \triangleq s_0 \in S_0$, then $a\#Z_0 \in \Gamma_0$ for an $a \in A$ iff $\nabla \vdash a\#t_0$ and $\nabla \vdash a\#s_0$. Notice that requiring this property does not imply a lose of generality: our algorithm starts with no equation in the store, and each time an equation is moved to the store the Sol rule adds the required freshness constraints (by inspection of Sol). Moreover, freshness constraints are only removed from the freshness context when Mer removes the corresponding equation from the store (by inspection of Mer). Then for any $Z_0 : t_0 \triangleq s_0 \in P_0 \cup S_0$ we have $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1\cdots\vartheta_n\rangle \preceq \langle \nabla, t_0\rangle$ and $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1\cdots\vartheta_n\rangle \preceq \langle \nabla, s_0\rangle$.

Assume the statement is true for any derivation of the length $l < n$ and prove it for a derivation $P_0; S_0; \Gamma_0; \vartheta_0 \Longrightarrow^{+} \emptyset; S_n; \Gamma_n; \vartheta_0\vartheta_1\cdots\vartheta_n$ of the length $n$.

Below the composition $\vartheta_i\vartheta_{i+1}\cdots\vartheta_k$ is abbreviated as $\vartheta_i^k$ with $k \geq i$.

Let $Z_0 : t_0 \triangleq s_0$ be an AUE selected for transformation from $P_0 \cup S_0$. We consider each rule:

Dec: $Z_0 = X$, $t_0 = \mathsf{h}(t_1, \ldots, t_m)$, $s_0 = \mathsf{h}(s_1, \ldots, s_m)$, $\Gamma_1 = \Gamma_0$, and $\vartheta_1 = \{X \mapsto \mathsf{h}(Y_1, \ldots, Y_m)\}$. By the induction hypothesis (IH), $\langle \Gamma_n \setminus \Gamma_1, Y_i\vartheta_2^n\rangle \preceq \langle \nabla, t_i\rangle$ and $\langle \Gamma_n \setminus \Gamma_1, Y_i\vartheta_2^n\rangle \preceq \langle \nabla, s_i\rangle$ for all $1 \leq i \leq m$. Hence by definition of $\preceq$ for terms-in-context, there exist substitutions $\sigma$ and $\varphi$ such that:

- $\vartheta_2^n\sigma$ and $\vartheta_2^n\varphi$ respect $\Gamma_n \setminus \Gamma_1$,
- $(\Gamma_n \setminus \Gamma_1)\sigma \subseteq \nabla$ and $(\Gamma_n \setminus \Gamma_1)\varphi \subseteq \nabla$, and
- $\nabla \vdash Y_i\vartheta_2^n\sigma \approx t_i$ and $\nabla \vdash Y_i\vartheta_2^n\varphi \approx s_i$ for all $1 \leq i \leq m$.

Finally, since $\vartheta_1 = \{X \mapsto \mathsf{h}(Y_1, \ldots, Y_m)\}$ and $\Gamma_n \setminus \Gamma_0 = \Gamma_n \setminus \Gamma_1$, we obtain $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$ and $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, s_0\rangle$.

Abs: $Z_0 = X$, $t_0 = a.t$, $s_0 = b.s$, $\Gamma_1 = \Gamma_0$, and $\vartheta_1 = \{X \mapsto c.Y\}$, where $\nabla \vdash c\#a.t$ and $\nabla \vdash c\#b.s$. $P_1$ contains the AUE $Y : (c\ a)\cdot t \triangleq (c\ b)\cdot s$. By the IH, $\langle \Gamma_n \setminus \Gamma_1, Y_i\vartheta_2^n\rangle \preceq \langle \nabla, (c\ a)\cdot t\rangle$ and $\langle \Gamma_n \setminus \Gamma_1, Y_i\vartheta_2^n\rangle \preceq \langle \nabla, (c\ b)\cdot s\rangle$ hence $\nabla \vdash Y\vartheta_2^n\sigma \approx (c\ a)\cdot t$ and $\nabla \vdash Y\vartheta_2^n\varphi \approx (c\ b)\cdot s$ for some $\sigma$ and $\varphi$ that in addition satisfy the other properties (as above) for $\preceq$. Then, since we also have that $\nabla \vdash c\#a.t$ and $\nabla \vdash c\#b.s$ we can prove, with the $\approx$-abs rules, that $\nabla \vdash c.Y\vartheta_2^n\sigma \approx a.t$ and $\nabla \vdash c.Y\vartheta_2^n\varphi \approx b.s$. Finally, since $\vartheta_1 = \{X \mapsto c.Y\}$ and $\Gamma_n \setminus \Gamma_0 = \Gamma_n \setminus \Gamma_1$, we obtain $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$ and $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, s_0\rangle$.

Sol: $Z_0 = X$, $t_0 = t$, $s_0 = s$, $\Gamma_1 \setminus \Gamma_0 = \{a\#X \mid a \in A, \nabla \vdash a\#t, \text{ and } \nabla \vdash a\#s\}$ and $\vartheta_1 = \varepsilon$. By the IH we have that $(\Gamma_n \setminus \Gamma_1)\sigma \subseteq \nabla$, $(\Gamma_n \setminus \Gamma_1)\varphi \subseteq \nabla$, $\nabla \vdash X\vartheta_2^n\sigma \approx t$, and $\nabla \vdash X\vartheta_2^n\varphi \approx s$ for some $\sigma$ and $\varphi$ respecting $\Gamma_n \setminus \Gamma_1$.

Since $\vartheta_1 = \varepsilon$, from the IH we get $\nabla \vdash X\vartheta_1^n\sigma \approx t$. To show that $(\Gamma_n \setminus \Gamma_0)\sigma \subseteq \nabla$ take $a\#Y \in \Gamma_n \setminus \Gamma_0$ for some $a$.

- If $a\#Y \in \Gamma_n \setminus \Gamma_1$, then $\{a\#Y\}\sigma \subseteq \nabla$ by the IH,
- otherwise, if $a\#Y \notin \Gamma_n \setminus \Gamma_1$, then $a\#Y \in \Gamma_1 \setminus \Gamma_0$ with $X = Y$ and $X\vartheta_2^n = X$. By the IH, $\nabla \vdash X\sigma \approx t$, besides, we know $\nabla \vdash a\#t$. Therefore, we know $\nabla \vdash a\#X\sigma$, which by Theorem 1 implies $\{a\#X\}\sigma = \{a\#Y\}\sigma \subseteq \nabla$. Thus, $(\Gamma_n \setminus \Gamma_0)\sigma \subseteq \nabla$.

Hence, we proved $\langle \Gamma_n \setminus \Gamma_0, X\vartheta_1^n\rangle \preceq \langle \nabla, t\rangle$, which is the same as $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$. $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, s_0\rangle$ can be proved analogously.

Mer: First, we show that the following holds : For all $k \geq 0$, If $Z_k : t_k \triangleq s_k \in S_k$ and $c\#Z_k \in \Gamma_k$ for a $c \in A$, then $\nabla \vdash c\#t_k$ and $\nabla \vdash c\#s_k$.

Proceed by induction on $k$. If $k = 0$, then it follows from the assumption on $P_0; S_0, \Gamma_0; \vartheta_0$. Assume it is true for $k$ and show it for $k+1$. Take $Z_{k+1} : t_{k+1} \triangleq s_{k+1} \in S_{k+1}$. We have two alternatives: Either $Z_{k+1} : t_{k+1} \triangleq s_{k+1}$ has been a subject of the Mer rule at this step, or not. If not, then either it was already in $S_k$ or was introduced at this step. In either case, by IH or because it has been introduced with Sol rule, if $c\#Z_k \in \Gamma_k$ for a $c \in A$, then $\nabla \vdash c\#t_k$ and $\nabla \vdash c\#s_k$. If $Z_{k+1} : t_{k+1} \triangleq s_{k+1}$ was a subject of the Mer rule, then there exists some $U_k : r_k \triangleq q_k \in S_k$, such that $\nabla \vdash \pi_k\cdot t_{k+1} \approx r_k$, $\nabla \vdash \pi_k\cdot s_{k+1} \approx q_k$. Moreover, for all $d\#U_k \in S_k$ we now have $\pi_k^{-1}\cdot d\#Z_{k+1} \in S_{k+1}$, and all $c\#Z_{k+1} \in S_k$ are retained in $S_{k+1}$. For these $c$'s, since $Z_{k+1} : t_{k+1} \triangleq s_{k+1} \in S_k$, by the induction hypothesis we have $\nabla \vdash c\#t_{k+1}$ and $\nabla \vdash c\#s_{k+1}$. As for $\pi_k^{-1}\cdot d\#Z_{k+1} \in S_{k+1}$, here we need to show $\nabla \vdash \pi_k^{-1}\cdot d\#t_{k+1}$ and $\nabla \vdash \pi_k^{-1}\cdot d\#s_{k+1}$. By the induction hypothesis we know $\nabla \vdash d\#r_k$. Then $\nabla \vdash \pi_k^{-1}\cdot d\#\pi_k^{-1}\cdot r_k$ and since $\nabla \vdash \pi_k\cdot t_{k+1} \approx r_k$, we get $\nabla \vdash \pi_k^{-1}\cdot d\#t_{k+1}$. $\nabla \vdash \pi_k^{-1}\cdot d\#s_{k+1}$ can be shown similarly, using $\nabla \vdash d\#q_k$.

Now we turn to proving the Mer case itself. In this case, there exist $X : t_1 \triangleq s_1 \in S_0$, $Y : t_2 \triangleq s_2 \in S_0$, and $\pi$ such that $\nabla \vdash \pi\cdot t_1 \approx t_2$ and $\nabla \vdash \pi\cdot s_1 \approx s_2$. Moreover, by the construction of the derivation, $X : t_1 \triangleq s_1$ is either retained in $S_n$, or is removed from there because there exist an AUE $Z : t_n \triangleq s_n \in S_n$ and a permutation $\rho$ such that $\nabla \vdash \rho\cdot t_n \approx t_1$, $\nabla \vdash \rho\cdot t_n \approx s_1$, and $X\vartheta_1^n = \rho\cdot Z$. We can turn these two cases into one, permitting $Z = X$, $t_n = t_1$, $s_n = s_1$, and $\rho = Id$ to cover also the first case.

Therefore, we can say that there exists a AUE $Z : t_n \triangleq s_n \in S_n$ such that for some permutation $\rho$, $X\vartheta_1^n = X\vartheta_2^n = \rho\cdot Z$, $Y\vartheta_1^n = \pi\cdot X\vartheta_2^n = \pi\cdot\rho\cdot Z$, $\Gamma_n \setminus \Gamma_0 = \{(\pi\cdot\rho)^{-1}a\#Z \mid a\#Y \in \Gamma_0\}$, $\nabla \vdash \pi\cdot\rho\cdot t_n \approx t_2$, $\nabla \vdash \pi\cdot\rho\cdot s_n \approx s_2$, $\nabla \vdash \rho\cdot t_n \approx t_1$, and $\nabla \vdash \rho\cdot s_n \approx s_1$.

We want to prove $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$. First, we take $\sigma$ such that $Z\sigma = t_n$ and show $(\Gamma_n \setminus \Gamma_0)\sigma \subseteq \nabla$. For this, we try to prove $\{b\#U\}\sigma \subseteq \nabla$ for all $b\#U \in \Gamma_n \setminus \Gamma_0$. By the IH, we have $\{b\#U\}\sigma \subseteq \nabla$ for all $b\#U \in \Gamma_n \setminus \Gamma_1$. Note that

$Z\sigma = t_n$ does not restrict generality, because if $U = Z$, then by the proposition we proved at the beginning of the Mer case we have that $b\#U \in \Gamma_n \setminus \Gamma_1$ implies $\nabla \vdash b\#t_n$.

Therefore, $\{b\#U\}\sigma = \mathsf{FC}(\{b\#Z\sigma\}) = \mathsf{FC}(\{b\#t_n\})$ and by Theorem 1 we indeed have $\{b\#U\}\sigma \subseteq \nabla$. Now assume $b\#U \in (\Gamma_n \setminus \Gamma_0) \setminus (\Gamma_n \setminus \Gamma_1)$. Then $b\#U \in \Gamma_n \cap (\Gamma_1 \setminus \Gamma_0)$. That means, $b\#U = \pi^{-1}\cdot a\#X$, where $a\#Y \in \Gamma_0$. Moreover, the AUE $X : t_1 \stackrel{?}{=} s_1$ has been retained in $S_n$. From the latter we have, in fact, $Z = X$, $t_n = t_1$, and $s_n = s_1$. Then $\{b\#U\}\sigma = \mathsf{FC}(\{\pi^{-1}\cdot a\#X\sigma \mid a\#Y \in \Gamma_0\}) = \mathsf{FC}(\{a\#\pi\cdot t_1 \mid a\#Y \in \Gamma_0\})$. On the other hand, from the assumption on $P_0; S_0; \Gamma_0; \vartheta_0$ we know that for all $a\#Y \in \Gamma_0$ we have $\nabla \vdash a\#t_2$, from which by $\nabla \vdash \pi\cdot t_1 \approx t_2$ we get $\nabla \vdash a\#\pi\cdot t_1$. Hence, we can apply Theorem 1 to $\mathsf{FC}(\{a\#\pi\cdot t_1 \mid a\#Y \in \Gamma_0\})$, obtaining $\{b\#U\}\sigma \subseteq \nabla$ also in this case. Hence, $(\Gamma_n \setminus \Gamma_0)\sigma \subseteq \nabla$.

It remains to prove $\nabla \vdash Z_0\vartheta_1^n\sigma \approx t_0$. First, assume $Z_0 = X$, $t_0 = t_1$, $s_0 = s_1$. Then we have $\nabla \vdash Z_0\vartheta_2^n\sigma \approx t_0$, because $Z_0\vartheta_2^n\sigma = \rho\cdot Z\sigma = \rho\cdot t_n$ and we know that $\nabla \vdash \rho\cdot t_n \approx t_1$. Since $Z_0\vartheta_2^n = Z_0\vartheta_1^n$, we get $\nabla \vdash Z_0\vartheta_1^n\sigma \approx t_0$. Hence, we proved $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$ for this case. $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, s_0\rangle$ can be proved similarly.

Now let $Z_0 = Y$, $t_0 = t_2$, $s_0 = s_2$ and prove again $\nabla \vdash Z_0\vartheta_1^n\sigma \approx t_0$. Then $Z_0\vartheta_1^n\sigma = \pi\cdot\rho\cdot Z\sigma = \pi\cdot\rho\cdot t_n$. But we have already seen that $\nabla \vdash \pi\cdot\rho\cdot t_n \approx t_2$. Hence, $\nabla \vdash Z_0\vartheta_1^n\sigma \approx t_0$ is proved. It implies $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, t_0\rangle$ for this case. $\langle \Gamma_n \setminus \Gamma_0, Z_0\vartheta_1^n\rangle \preceq \langle \nabla, s_0\rangle$ can be proved similarly.

■

**Theorem 5** (Completeness of $\mathfrak{N}$)**.** *Given terms $t$ and $s$ and freshness contexts $\nabla$ and $\Gamma$, all based on a finite set of atoms $A$, if $\langle \Gamma, r\rangle$ is an $A$-based generalization of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, then there exists a derivation $\{X : t \stackrel{?}{=} s\}; \emptyset; \emptyset; \varepsilon \Longrightarrow^+ \emptyset; S; \Gamma'; \sigma$ obtained by an execution of $\mathfrak{N}$, such that $\langle \Gamma, r\rangle \preceq \langle \Gamma', X\sigma\rangle$.*

*Proof:* By structural induction on $r$. We can assume without loss of generality that $\langle \Gamma, r\rangle$ is an lgg of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$.

Let $r$ be an atom $a$. Then $t = s = a$. Therefore, the Dec rule gives $\langle \emptyset, a\rangle$ as the computed answer. To show that $\langle \Gamma, a\rangle \preceq \langle \emptyset, a\rangle$, it is enough to take a substitution $\sigma$ such that $X\sigma \neq b$ for each $b\#X \in \Gamma$. Note that it is not necessary $b \in A$.

Let $r$ be an abstraction $c.r'$. Then $t = a.t'$, $s = b.s'$, $c \in A$, $\nabla \vdash c\#t$, $\nabla \vdash c\#s$, and $\langle \Gamma, r'\rangle$ is an $A$-based generalization of $\langle \nabla, t'\rangle$ and $\langle \nabla, s'\rangle$. In this case, the Abs rule can be applied, which gives $\{Y : (c\,a)\cdot t' \stackrel{?}{=} (c\,b)\cdot s'\}; \emptyset; \emptyset; \sigma_1$, where $\sigma_1 = \{X \mapsto c.Y\}$. By the induction hypothesis, we can compute $\Gamma'$ and $\sigma_2$ such that $\langle \Gamma, r'\rangle \preceq \langle \Gamma', Y\sigma_2\rangle$. Let $\sigma = \sigma_1\sigma_2$. We get $\langle \Gamma, r\rangle = \langle \Gamma, c.r'\rangle \preceq \langle \Gamma', c.Y\sigma_2\rangle = \langle \Gamma', X\sigma\rangle$.

Let $r$ be a suspension $\pi\cdot Z$. Since $\langle \Gamma, r\rangle$ is an lgg of $\langle \nabla, t\rangle$ and $\langle \nabla, s\rangle$, the context $\Gamma$ contains all constraints $\pi^{-1}\cdot a\#Z$ such that $\nabla \vdash a\#t$ and $\nabla \vdash a\#s$, and the following alternatives are possible:

(a) $t$ and $s$ have distinct heads: $\mathrm{Head}(t) \neq \mathrm{Head}(s)$, or
(b) $t$ and $s$ are both suspensions: $t = \pi_1\cdot Y_1$ and $s = \pi_2\cdot Y_2$, where $\pi_1, \pi_2$ and $Y_1, Y_2$ are not necessarily distinct, or
(c) $t$ and $s$ are abstractions, but $A$ does not contain an appropriate fresh atom to uniformly rename the bound atoms in $t$ and $s$.

These alternatives give exactly the conditions of the Sol rule. Hence, we can apply it, getting $\emptyset; \{X : t \stackrel{?}{=} s\}; \Gamma'; \sigma$, where $\Gamma' = \{a\#X \mid a \in A \wedge \nabla \vdash a\#t \wedge \nabla \vdash a\#s\}$ and $\sigma = \varepsilon$. Then $\langle \Gamma, r\rangle \preceq \langle \Gamma', X\sigma\rangle$, which can be confirmed by the substitution $\{Z \mapsto \pi^{-1}\cdot X\}$.

Let $r$ be a term $f(r_1, \ldots, r_n)$. Then $t = f(t_1, \ldots, t_n)$, $s = f(s_1, \ldots, s_n)$, and $\langle \Gamma, r_i\rangle$ is a generalization of $\langle \nabla, t_i\rangle$ and $\langle \nabla, s_i\rangle$. We proceed by the Dec rule, obtaining $\{Y_i : t_i \stackrel{?}{=} s_i \mid 1 \le i \le n\}; \emptyset; \emptyset; \{X \mapsto f(Y_1, \ldots, Y_n)\}$. By the induction hypothesis, we can construct derivations $D_1, \ldots, D_n$ computing the substitutions $\sigma_1, \ldots, \sigma_n$, respectively, such that $\langle \Gamma, r_i\rangle \preceq \langle \Gamma_i', Y_i\sigma_i\rangle$ for $1 \le i \le n$. We combine these derivations, together with the initial Dec step, into one derivation of the form $\mathsf{D} = \{X : t \stackrel{?}{=} s\}; S_0; \Gamma_0'; \sigma_0 \Longrightarrow \{Y_i : t_i \stackrel{?}{=} s_i \mid 1 \le i \le n\}; S_1; \Gamma_1'; \sigma_0\sigma_1 \Longrightarrow^* \emptyset; S_n; \Gamma_n'; \sigma_0\sigma_1\cdots\sigma_n$, where $\Gamma_0' = \Gamma_1' = \emptyset$, $\sigma_0 = \varepsilon$, and $\sigma_1 = \{X \mapsto f(Y_1, \ldots, Y_n)\}$. If $r$ does not contain the same variable more than once, $\langle \Gamma, r_i\rangle \preceq \langle \Gamma_i', Y_i\sigma_i\rangle$ for all $1 \le i \le n$ imply $\langle \Gamma, r\rangle = \langle \Gamma, f(r_1, \ldots, r_n)\rangle \preceq \langle \Gamma', f(Y_1, \ldots, Y_n)\rangle = \langle \Gamma', X\sigma\rangle$. If $r$ contains the same variable at positions $\mathsf{p}_1$ and $\mathsf{p}_2$ (in subterms of the form $\pi_1\cdot Z$ and $\pi_2\cdot Z$), it indicates that

(a) the path to $\mathsf{p}_1$ is the same (modulo bound atom renaming) in $t$ and $s$. It equals (modulo bound atom renaming) the path to $\mathsf{p}_1$ in $r$, and
(b) the path to $\mathsf{p}_2$ is the same (modulo bound atom renaming) in $t$ and $s$. It equals (modulo bound atom renaming) the path to $\mathsf{p}_2$ in $r$.
(c) there exists a substitution $\vartheta_1$, which respects $\Gamma$, such that $\Gamma \vdash \pi_1\cdot Z\vartheta_1 \approx \tau_1\cdot t|_{\mathsf{p}_1}$ and $\Gamma \vdash \pi_2\cdot Z\vartheta_1 \approx \tau_2\cdot t|_{\mathsf{p}_2}$, where $\tau_1$ and $\tau_2$ are permutations which rename atoms bound in $t$ by fresh ones,
(d) there exists a substitution $\vartheta_2$, which respects $\Gamma$, such that $\Gamma \vdash \pi_1\cdot Z\vartheta_2 \approx \rho_1\cdot s|_{\mathsf{p}_1}$ and $\Gamma \vdash \pi_2\cdot Z\vartheta_2 \approx \rho_2\cdot s|_{\mathsf{p}_2}$, where $\rho_1$ and $\rho_2$ are permutations which rename atoms bound in $s$ by fresh ones,

Then, because of (a) and (b), we should have two AUEs in $S_n$: One, between (renamed variants of) $t|_{\mathsf{p}_1}$ and $s|_{\mathsf{p}_1}$, and the other one between (renamed variants of) $t|_{\mathsf{p}_2}$ and $s|_{\mathsf{p}_2}$. The possible renaming of bound atoms is caused by the fact that Abs might have been applied to obtain the AUEs. From (c) and (d) we know that $\tau_1, \tau_2, \rho_1, \rho_2$ are the names of those renaming permutations. Let those AUEs be $Z_1 : \tau_1\cdot t|_{\mathsf{p}_1} \stackrel{?}{=} \rho_1\cdot s|_{\mathsf{p}_1}$ and $Z_2 : \tau_2\cdot t|_{\mathsf{p}_2} \stackrel{?}{=} \rho_2\cdot s|_{\mathsf{p}_2}$.

From (c) we get $\Gamma \vdash Z\vartheta_1 \approx \pi_1^{-1}\tau_1\cdot t|_{\mathsf{p}_1}$ and $\Gamma \vdash Z\vartheta_1 \approx \pi_2^{-1}\tau_2\cdot t|_{\mathsf{p}_2}$, which imply $\Gamma \vdash \pi_1^{-1}\tau_1\cdot t|_{\mathsf{p}_1} \approx \pi_2^{-1}\tau_2\cdot t|_{\mathsf{p}_2}$ and, finally, $\Gamma \vdash \pi_2\pi_1^{-1}\cdot\tau_1\cdot t|_{\mathsf{p}_1} \approx \tau_2\cdot t|_{\mathsf{p}_2}$. Similarly, from (d) we get $\Gamma \vdash \pi_2\pi_1^{-1}\rho_1\cdot s|_{\mathsf{p}_1} \approx \rho_2\cdot s|_{\mathsf{p}_2}$.

That means, we can make the step with the Mer rule for $Z_1 : \tau_1\cdot t|_{\mathsf{p}_1} \stackrel{?}{=} \rho_1\cdot s|_{\mathsf{p}_1}$ and $Z_2 : \tau_2\cdot t|_{\mathsf{p}_2} \stackrel{?}{=} \rho_2\cdot s|_{\mathsf{p}_2}$ with the substitution $\sigma_1' = \{Z_2 \mapsto \pi_2\pi_1^{-1}\cdot Z_1\}$. We can repeat this process for all duplicated variables in $r$, extending $\mathsf{D}$ to the derivation $\{X : t \stackrel{?}{=} s\}; S_0; \Gamma_0'; \sigma_0 \Longrightarrow \{Y_i : t_i \stackrel{?}{=} s_i \mid$

$1 \leq i \leq n\}$; $S_1; \Gamma'_1; \sigma_0 \Longrightarrow^* \emptyset; S_n; \Gamma'_n; \sigma_0\sigma_1\cdots\sigma_n \Longrightarrow^+ \emptyset; S_{n+m}; \Gamma'_{n+m}; \sigma_0\sigma_1\cdots\sigma_n\sigma'_1\cdots\sigma'_m$, where $\sigma'_1, \ldots, \sigma'_m$ are substitutions introduced by the applications of the Mer rule. Let $\sigma = \sigma_0\sigma_1\cdots\sigma_n\sigma'_1\cdots\sigma'_m$ and $\Gamma' = \Gamma'_{n+m}$. By this construction, we have $\langle \Gamma, r\rangle \preceq \langle \Gamma', X\sigma\rangle$, which finishes the proof. ∎

**Theorem 6** (Uniqueness Modulo $\simeq$)**.** *Let $t$ and $s$ be terms and $\nabla$ be a freshness context that are based on the same finite set of atoms. Let $\{X : t \triangleq s\}; \emptyset; \emptyset;\ \varepsilon \Longrightarrow^+ \emptyset;\ S_1;\ \Gamma_1;\ \sigma_1$ and $\{X : t \triangleq s\}; \emptyset; \emptyset;\ \varepsilon \Longrightarrow^+ \emptyset;\ S_2;\ \Gamma_2;\ \sigma_2$ be two maximal derivations in $\mathfrak{N}$. Then $\langle \Gamma_1, X\sigma_1\rangle \simeq \langle \Gamma_2, X\sigma_2\rangle$.*

*Proof:* It is not hard to notice that if it is possible to change the order of applications of rules (but sticking to the same selected AUEs for each rule) then the result remains the same (modulo fresh variable and atom names): Let $\mathsf{D}_1$ and $\mathsf{D}_2$ be two two-step derivations $\mathsf{D}_1 = P_1; S_1; \Gamma_1; \sigma_1 \Longrightarrow_{\mathsf{R}_1} P_2; S_2; \Gamma_2; \sigma_1\vartheta_1 \Longrightarrow_{\mathsf{R}_2} P_3; S_3; \Gamma_3; \sigma_1\vartheta_1\vartheta_2$ and $\mathsf{D}_2 = P_1; S_1; \Gamma_1; \sigma_1 \Longrightarrow_{\mathsf{R}_2} P'_2; S'_2; \Gamma'_2; \sigma_1\vartheta_2 \Longrightarrow_{\mathsf{R}_1} P'_3; S'_3; \Gamma'_3; \sigma_1\vartheta_2\vartheta_1$, where $\mathsf{R}_1$ and $\mathsf{R}_2$ are (not necessarily different) rules and each of them transforms *exactly the same AUE(s)* in both $\mathsf{D}_1$ and $\mathsf{D}_2$. Then these AUE(s) are already present in $P_1 \cup S_1$: They are introduced neither by $\mathsf{R}_1$ nor by $\mathsf{R}_2$. Therefore, $\mathrm{Dom}(\vartheta_2) \cap \mathrm{Ran}(\vartheta_1) = \mathrm{Dom}(\vartheta_1) \cap \mathrm{Ran}(\vartheta_2) = \emptyset$. Moreover, if we assume that the fresh variables and atoms introduced by the rules are the same in both derivations, then $P_3 = P'_3$, $S_3 = S'_3$, $\Gamma_3 = \Gamma'_3$, and $\sigma_1\vartheta_1\vartheta_2 = \sigma_1\vartheta_2\vartheta_1$.

Decomposition, Abstraction, and Solving rules transform the selected AUE in a unique way. We show that it is irrelevant in which order we decide equivariance in the Merging rule.

Let $P;\ \{Z : t_1 \triangleq s_1, Y : t_2 \triangleq s_2\} \uplus S;\ \Gamma;\ \sigma \Longrightarrow P;\ \{Z : t_1 \triangleq s_1\} \uplus S;\ \Gamma\{Y \mapsto \pi\cdot Z\};\ \sigma\{Y \mapsto \pi\cdot Z\}$ be the merging step with $\nabla \vdash \pi\cdot t_1 \approx t_2$ and $\nabla \vdash \pi\cdot s_1 \approx s_2$. If we do it in the other way around, we would get the step $P;\ \{Z : t_1 \triangleq s_1, Y : t_2 \triangleq s_2\} \uplus S;\ \Gamma;\ \sigma \Longrightarrow P;\ \{Y : t_2 \triangleq s_2\} \uplus S;\ \Gamma\{Z \mapsto \pi^{-1}\cdot Y\};\ \sigma\{Z \mapsto \pi^{-1}\cdot Y\}$.

Let $\vartheta_1 = \sigma\varphi_1$ with $\varphi_1 = \{Y \mapsto \pi\cdot Z\}$ and $\vartheta_2 = \sigma\varphi_2$ with $\varphi_2 = \{Z \mapsto \pi^{-1}\cdot Y\}$. Our goal is to prove that $\langle \Gamma\varphi_1, X\vartheta_1\rangle \simeq \langle \Gamma\varphi_2, X\vartheta_2\rangle$. For this, we need to prove both $\langle \Gamma\varphi_1, X\vartheta_1\rangle \preceq \langle \Gamma\varphi_2, X\vartheta_2\rangle$ and $\langle \Gamma\varphi_2, X\vartheta_2\rangle \preceq \langle \Gamma\varphi_1, X\vartheta_1\rangle$.

First, prove $\langle \Gamma\varphi_1, X\vartheta_1\rangle \preceq \langle \Gamma\varphi_2, X\vartheta_2\rangle$. We should find such a $\varphi$ that $\Gamma\varphi_1\varphi \subseteq \Gamma\varphi_2$ and $\Gamma\varphi_2 \vdash X\vartheta_1\varphi \approx X\vartheta_2$.

Take $\varphi = \varphi_2$. Note that $\varphi_1\varphi_2 = \varphi_2$, because $\pi\pi^{-1}\cdot Y = Y$. Therefore $X\vartheta_1\varphi = X\sigma\varphi_1\varphi_2 = X\sigma\varphi_2 = X\vartheta_2$ and $\Gamma\varphi_2 \vdash X\vartheta_1\varphi \approx X\vartheta_2$ holds.

As for $\Gamma\varphi_1\varphi \subseteq \Gamma\varphi_2$, note that $\varphi_2$ respects $\Gamma\varphi_1$, because it replaces a variable with a suspension and the FC algorithm will have to apply only Sus-E rule. We introduce notations $\Gamma_U$ and $\overline{\Gamma}_U$ for any freshness context $\Gamma$ and a variable $U$, denoting $\Gamma_U := \{a\#U \mid a\#U \in \Gamma\}$ and $\overline{\Gamma}_U := \Gamma \setminus \Gamma_U$. Then $\Gamma\varphi_1 = \overline{\Gamma}_Y \cup \Gamma_Y\varphi_1$ and $\Gamma\varphi_2 = \overline{\Gamma}_Z \cup \Gamma_Z\varphi_2$.

Under this notation, $\Gamma\varphi_1\varphi_2 = \overline{\Gamma}_Y\varphi_2 \cup \Gamma_Y\varphi_1\varphi_2$. Take $\overline{\Gamma}_Y\varphi_2$. We have $\overline{\Gamma}_Y\varphi_2 = (\overline{\Gamma}_Y \setminus (\overline{\Gamma}_Y)_Z) \cup ((\overline{\Gamma}_Y)_Z)\varphi_2 = (\overline{\Gamma}_Y \setminus \Gamma_Z) \cup \Gamma_Z\varphi_2$. Since $\Gamma_Z \cap \Gamma_Z\varphi_2 = \emptyset$, the we obtain $(\overline{\Gamma}_Y \setminus \Gamma_Z) \cup \Gamma_Z\varphi_2 = (\overline{\Gamma}_Y \cup \Gamma_Z\varphi_2) \setminus \Gamma_Z$. As for $\Gamma_Y\varphi_1\varphi_2$, it is easy to see that $\Gamma_Y\varphi_1\varphi_2 = \Gamma_Y$.

Hence, we get $\Gamma\varphi_1\varphi_2 = ((\overline{\Gamma}_Y \cup \Gamma_Z\varphi_2) \setminus \Gamma_Z) \cup \Gamma_Y$. Since $\Gamma_Z \cap \Gamma_Y = \emptyset$, we get $((\overline{\Gamma}_Y \cup \Gamma_Z\varphi_2) \setminus \Gamma_Z) \cup \Gamma_Y = ((\overline{\Gamma}_Y \cup \Gamma_Z\varphi_2) \cup \Gamma_Y) \setminus \Gamma_Z = (\Gamma \cup \Gamma_Z\varphi_2) \setminus \Gamma_Z$. Since $\Gamma_Z\varphi_2 \cap \Gamma_Z = \emptyset$, we get $(\Gamma \cup \Gamma_Z\varphi_2) \setminus \Gamma_Z = (\Gamma \setminus \Gamma_Z) \cup \Gamma_Z\varphi_2$. Hence, $\Gamma\varphi_1\varphi_2 = \overline{\Gamma}_Z \cup \Gamma_Z\varphi_2 = \Gamma\varphi_2$.

We proved $\langle \Gamma\varphi_1, X\vartheta_1\rangle \preceq \langle \Gamma\varphi_2, X\vartheta_2\rangle$. With a similar reasoning we can show $\langle \Gamma\varphi_2, X\vartheta_2\rangle \preceq \langle \Gamma\varphi_1, X\vartheta_1\rangle$. ∎

**Theorem 8** (Soundness of $\mathfrak{E}$)**.** *Let $\{t \approx s\};\ \nabla;\ A;\ Id \Longrightarrow^* \emptyset;\ \nabla;\ B;\ \pi$ be a derivation in $\mathfrak{E}$, then $\pi$ is an $A$-based permutation such that $\nabla \vdash \pi\cdot t \approx s$.*

*Proof:* We assume the success state with $\pi$ being the computed permutation. Since Sol-E is the only rule which adds a new swapping to $\pi$ and the swapped atoms are required to be from $A$, $\pi$ is $A$-based.

The proof is by induction on the length of the derivation, and then, by case analysis on the applied rule. Let $\Gamma$ be the freshness environment containing all statements $\acute{c}\#X$ form by a fresh atom $\acute{c}$ introduced along all the derivation and a variable $X$ of the initial equation.

For any transformation step $E;\ \nabla;\ A;\ \pi \Longrightarrow E';\ \nabla;\ A';\ \pi'$ we will prove that if $\nabla \cup \Gamma \vdash \pi'\cdot t'_i \approx s'_i$, for any $t'_i \approx s'_i \in E'$, then $\nabla \cup \Gamma \vdash \pi\cdot t_i \approx s_i$ for any $t_i \approx s_i \in E$, for any possible applied rule. By induction, we will have $\nabla \cup \Gamma \vdash \pi\cdot t \approx s$ for the initial equivariance equation $t \approx s$. Since $\Gamma$ is not relevant to prove $t \approx s$, we have also $\nabla \vdash \pi\cdot t \approx s$.

Soundness of Dec-E: From $\nabla \vdash \pi\cdot t_1 \approx s_1, \ldots, \nabla \vdash \pi\cdot t_n \approx s_n$, follows directly, by the theory of alpha-equivalence $\nabla \vdash f(\pi\cdot t_1, \cdots, \pi\cdot t_n) \approx f(s_1, \cdots, s_n)$ and by the rule of swapping application $\pi\cdot f(t_1, \cdots, t_n) = f(\pi\cdot t_1, \cdots, \pi\cdot t_n)$, that $\nabla \vdash \pi\cdot f(t_1, \cdots, t_n) \approx f(s_1, \cdots, s_n)$. In this case the permutation, the set of atoms and the freshness context are not transformed by the rule.

Soundness of Alp-E: Let $\nabla$ be a freshness context containing $\Gamma$, in particular $\acute{c}\#X$ for any variable $X \in \mathrm{Vars}(t, s)$. Assume $\nabla \vdash \pi(a\,\acute{c})\cdot t \approx (b\,\acute{c})\cdot s$ by induction hypothesis. From this, using $\approx$-abs-1 and the fact that $\pi$ does not affect to $\acute{c}$, we can deduce $\nabla \vdash \pi\cdot c.(a\,\acute{c})\cdot t \approx c.(b\,\acute{c})\cdot s$. We can also construct a proof for $\nabla \vdash \acute{c}\#t$ and $\nabla \vdash \acute{c}\#s$. Therefore, using $\approx$-abs-2, we can deduce $\nabla \vdash \acute{c}.(a\,\acute{c})\cdot t \approx a.t$ and $\nabla \vdash \acute{c}.(b\,\acute{c})\cdot s \approx b.s$. Now using the lemmas about the transitivity of $\approx$ and additivity of permutation application:

$$\begin{array}{l} \textit{If } \nabla \vdash t \approx s \textit{ and } \nabla \vdash s \approx u \textit{ then } \nabla \vdash t \approx u \\ \textit{If } \nabla \vdash t \approx s \textit{ then } \nabla \vdash \pi\cdot t \approx \pi\cdot s \end{array}$$

we can deduce $\nabla \vdash \pi\cdot(a.t) \approx b.s$. This proof proves the soundness of Alp-E. Notice that $\pi$ does not change in this rule.

Soundness of Sus-E: By induction hypothesis, assume $\nabla \vdash \pi\,\pi_1\cdot a \approx \pi_2\cdot a$, for any atom $a$ such that $a \in A$ and $a\#X \notin \nabla$. Assume also $\Gamma \subset \nabla$, hence, for all fresh atoms, we have $\acute{c}\#X \in \nabla$. The rest of atoms $b$ are not fresh and satisfy $b \notin A$ and $b\#X \notin \nabla$. Since $\pi_1$ and $\pi_2$ only affect to atoms from

$A$ or fresh[4], and $\pi$ is $A$-based, we have $\pi\,\pi_1{\cdot}b = \pi_2{\cdot}b = b$. Therefore, $\nabla \vdash \pi\,\pi_1{\cdot}a \approx \pi_2{\cdot}a$, for any atom $a\#X \notin \nabla$, and by $\approx$-susp we deduce $\nabla \vdash \pi\,\pi_1{\cdot}X \approx \pi_2{\cdot}X$.

In the second phase we have to take into account that, in all derivations of the form $E;\ \nabla;\ A;\ \pi \Longrightarrow^* \emptyset;\ \nabla;\ B;\ \pi'\pi$, permutation $\pi'$ only affects to atoms from $A$. This can be proved by inspection of the rules.

Soundness of Rem-E: Let be the complete derivation as follows

$$\begin{array}{l}\{a \approx b\}\cup E;\ \nabla;\ A;\ \pi \Longrightarrow\\ E;\ \nabla;\ A \setminus \{b\};\ \pi \Longrightarrow^*\\ \emptyset;\ \nabla;\ B;\ \pi'\pi\end{array}$$

By induction hypothesis, $\pi'\pi$ solves $E$. Since the rule has been applied we also have $\pi{\cdot}a = b$. Now, the property above proves $\pi'{\cdot}b = b$, since $b \notin A \setminus \{b\}$. Therefore $\pi'\pi{\cdot}a = b$.

Soundness of Sol-E: let the derivation be:

$$\begin{array}{l}\{a \approx b\}\cup E;\ \nabla;\ A;\ \pi \Longrightarrow\\ E;\ \nabla;\ A \setminus \{b\};\ (\pi{\cdot}a\ b)\pi \Longrightarrow^*\\ \emptyset;\ \nabla;\ B;\ \pi'(\pi{\cdot}a\ b)\pi\end{array}$$

By induction hypothesis, $\pi'(\pi{\cdot}a\ b)\pi$ solves $E$. Since $b \notin A \setminus \{b\}$, we have $\pi'{\cdot}b = b$. Hence $\pi'(\pi{\cdot}a\ b)\pi{\cdot}a = \pi'{\cdot}b = b$, and the computed permutation also solves the equivariance equation $a \approx b$. ∎

**Theorem 9** (Completeness of $\mathfrak{E}$)**.** *Let $A$ be a finite set of atoms, $t, s$ be $A$-based terms, and $\nabla$ be a $A$-based freshness context. If $\nabla \vdash \mu{\cdot}t \approx s$ holds for some $A$-based permutation $\mu$, then there exists a derivation $\{t \approx s\};\ \nabla;\ A;\ Id \Longrightarrow^* \emptyset;\ \Gamma;\ B;\ \pi$, obtained by an execution of $\mathfrak{E}$, such that $\pi{\cdot}a = \mu{\cdot}a$ for any atom $a \in \mathrm{FA}(t)$.*

*Proof:* First show that under the conditions of the theorem, if $\{t \approx s\};\ \nabla;\ A;\ Id \Longrightarrow^* \emptyset;\ \Gamma;\ B;\ \pi$ is a derivation obtained by an execution of $\mathfrak{E}$, then $\pi{\cdot}a = \mu{\cdot}a$, for any atom $a \in \mathrm{FA}(t)$. Afterwards we prove that (under the conditions of the theorem) there is no failing derivation with the rules of $\mathfrak{E}$ starting from $\{t \approx s\};\ \nabla;\ A;\ Id$. Since all derivations are finite, it will imply the existence of $\{t \approx s\};\ \nabla;\ A;\ Id \Longrightarrow^* \emptyset;\ \Gamma;\ B;\ \pi$.

Let $\{t \approx s\};\nabla;A;Id \Longrightarrow^* E';\Gamma';B';\pi' \Longrightarrow^* \emptyset;\Gamma;B;\pi$ be a derivation, where $E';\ \Gamma';\ B';\ \pi'$ is the first system in the second phase of the algorithm. It means that $E'$ contains equations between atoms only, and the atoms of $t$ (except, maybe, some bound ones which disappear after the application of the Alp-E rule) appear in the left hand sides of equations in $E$. By Lemma 6, $\Gamma' \vdash \mu{\cdot}a_1 \approx a_2$, for all $a_1 \approx a_2 \in E'$. By Theorem 8 and Lemma 6 the same is true for $\pi$. Therefore, $\Gamma' \vdash \mu{\cdot}a_1 \approx \pi{\cdot}a_1$, for all $a_1 \approx a_2 \in E'$. For atoms, $\nabla \vdash a \approx b$ iff $a = b$. Hence, we get $\mu{\cdot}a_1 = \pi{\cdot}a_1$, for all $a_1 \in S$, where $\mathrm{FA}(t) \subseteq S \subseteq \mathrm{Atoms}(t)$. It proves $\pi{\cdot}a = \mu{\cdot}a$, for all $a \in \mathrm{FA}(t)$, when the desired successful derivation exists.

Now we show that no derivation with the rules of $\mathfrak{E}$ starting from $\{t \approx s\};\ \nabla;\ A;\ Id$ fails. Assume by contradiction that there exists such a failing derivation. Let $E';\nabla';A';\pi'$ be the final system in it, to which no rule applies. Analyzing the rules in $\mathfrak{E}$, one can easily conclude that it can be caused by one of the following two cases:

1) $E'$ contains an equivariance equation of the form $f(t_1,\ldots,t_n) \approx g(s_1,\ldots,s_m)$, where $f \neq g$.
2) $E'$ contains an equivariance equation of the form $a \approx b$, where $\pi'{\cdot}a \neq b$, such that $\pi'{\cdot}a \notin A'$ or $b \notin A'$.

In the first case, by Lemma 6 $\nabla \vdash \mu{\cdot}f(t_1,\ldots,t_n) \approx g(s_1,\ldots,s_m)$ should hold, but $f \neq g$ forbids it. Hence, this case is impossible.

Now we analyze the second case. Consider each condition.

Condition 1: $\pi'{\cdot}a \notin A'$. Then either $\pi'{\cdot}a$ is a fresh atom, or $\pi'{\cdot}a \in A \setminus A'$.

- *$\pi'{\cdot}a$ is a fresh atom:* Since $\pi'$ does not affect fresh atoms, we get $a \neq b$. On the other hand, we have $\Gamma' \vdash \mu{\cdot}a \approx b$ and, hence, $\mu{\cdot}a = b$, because $\mu{\cdot}a$ and $b$ are atoms. Since $\mu$ is $A$-based, $b \notin A$ implies $a = b$. A contradiction.
- *$\pi'{\cdot}a \in A \setminus A'$:* By Lemma 6 we get $\mu^{-1}\pi'{\cdot}a = \pi'^{-1}\pi'{\cdot}a = a$. Therefore, $\pi'{\cdot}a = \mu{\cdot}a$ and we get $\mu{\cdot}a \neq b$, which contradicts $\Gamma' \vdash \mu{\cdot}a \approx b$, because $\mu{\cdot}a$ and $b$ are atoms.

Condition 2: $b \notin A'$. Then either $b$ is a fresh atom, or $b \in A \setminus A'$.

- *$b$ is a fresh atom:* We obtain a contradiction by a reasoning similar to the case when $\pi'{\cdot}a$ is a fresh atom.
- *$b \in A \setminus A'$:* The atom $b$ has been removed from the set of atoms in the derivation earlier either at Sus-E, Rem-E, or Sol-E step, which indicates that there is $c \in A \setminus A'$ such that $c = \pi'^{-1}{\cdot}b$. Moreover, $c \neq a$. From Lemma 6 we get $c = \mu^{-1}{\cdot}b$ which, together with $c \neq a$, implies $\mu{\cdot}a \neq b$. But it contradicts $\Gamma' \vdash \mu{\cdot}a \approx b$.

The obtained contradiction proves that no derivation with the rules of $\mathfrak{E}$ starting from $\{t \approx s\};\ \nabla;\ A;\ Id$ fails. ∎

[4]Notice that $\pi_1$ and $\pi_2$ can only contain swappings of the original equation (i.e. $A$-based) and swappings introduced by Alp-E.