Fixed-Point Constraints for Nominal Equational Unification

Mauricio Ayala-Rincón[1]
Departments of Mathematics and Computer Science, Universidade de Brasília, Brasília, Brazil

Maribel Fernández
Department of Informatics, King's College London, London, UK

Daniele Nantes-Sobrinho[2]
Departments of Mathematics and Computer Science, Universidade de Brasília, Brasília, Brazil

Abstract

We propose a new axiomatisation of the alpha-equivalence relation for nominal terms, based on a primitive notion of fixed-point constraint. We show that the standard freshness relation between atoms and terms can be derived from the more primitive notion of permutation fixed-point, and use this result to prove the correctness of the new alpha-equivalence axiomatisation. This gives rise to a new notion of nominal unification, where solutions for unification problems are pairs of a fixed-point context and a substitution. Although it may seem less natural than the standard notion of nominal unifier based on freshness constraints, the notion of unifier based on fixed-point constraints behaves better when equational theories are considered: for example, nominal unification remains finitary in the presence of commutativity, whereas it becomes infinitary when unifiers are expressed using freshness contexts.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting, Theory of computation → Lambda calculus, Theory of computation → Algebraic semantics

Keywords and phrases nominal terms, fixed-point equations, nominal unification, equational theories

Digital Object Identifier 10.4230/LIPIcs.FSCD.2018.7

1 Introduction

This paper presents a new axiomatisation of α-equivalence for nominal terms via permutation fixed points, and revisits nominal unification in this setting.

In nominal syntax [16], *atoms* are used to represent object-level variables and *atom permutations* to implement renamings, following the nominal-sets approach advocated by Gabbay and Pitts [10, 12, 14]. Atoms can be abstracted over terms, the syntax $[a]s$ represents the abstraction of a in s. To rename an abstracted atom a to b, a *swapping* permutation $\pi = (a\,b)$ is applied. Thus, the action of π over $[a]s$, written as $(a\,b) \cdot [a]s$, produces the nominal term $[b]s'$, where s' is the result of replacing all occurrences of a in s by b, and all occurences of b in s by a. The α-equivalence relation between nominal terms is specified using swappings together with a *freshness relation* between atoms and terms, written $b\#s$, which roughly corresponds to b not occurring free in s.

In this setting, checking α-equivalence requires another first-order specialised calculus to check freshness constraints. For instance, checking whether $[a]s \approx_\alpha [b]t$ reduces to checking

[1] Author partially funded by CNPq 307672/2017-4.
[2] Author partially supported by FAP-DF 0193.001381/2017.

whether $s \approx_\alpha (b\,a) \cdot t$ and $a\#t$. The action of a permutation propagates down the structure of nominal terms, until a variable is reached: permutations suspend over variables. Thus, $\pi \cdot s$ represents the action of a permutation over a nominal term, but is not itself a nominal term unless s is a variable; for instance, $\pi \cdot X$ is a *suspension* (also called *moderated variable*), which is a nominal term.

The presence of moderated variables and atom-abstractions makes reasoning about equality of nominal terms more involved than in standard first-order syntax. For example, $\pi \cdot X \approx^?_\alpha \rho \cdot X$ is only true when X ranges over nominal terms, say s, for which all atoms in the difference set of π and ρ (i.e., the set $\{a : \pi(a) \neq \rho(a)\}$) are fresh in s.

If the support of a permutation π is fresh for X then $\pi \cdot X \approx_\alpha id \cdot X$. Thus a set of freshness constraints (i.e., a freshness context) can be used to specify that a permutation will have no effect on the instances of X. This is why in *nominal unification* [16], the solution for a problem is a pair consisting of a freshness context and a substitution.

The use of freshness contexts is natural when dealing with "syntactic" nominal unification, but in the presence of equational axioms (i.e., equational nominal unification) it is not straightforward. For example, in the case of C-nominal unification (nominal unification modulo commutativity), to specify that a permutation has no effect on the instances of X modulo C, in other words, to specify that the permutation does not affect a given C-equivalence class, we need something more than a freshness constraint (note that $(a\ b)(a+b) = b + a =_C a + b$, so the permutation $(a\ b)$ fixes the term $a + b$, despite the fact that a and b are not fresh).

In this paper, we propose to axiomatise α-equivalence of nominal terms using permutation fixed-point constraints: we write $\pi \curlywedge t$ (read "π fixes t") if t is a fixed-point of π. We show how to derive fixed-point constraints from primitive constraints of the form $\pi \curlywedge X$, and show the correctness of this approach by proving that the α-equivalence relation generated in this way coincides with the one axiomatised via freshness constraints. We then show how fixed-point constraints can be used to solve nominal unification problems modulo C.

In [4, 3, 2], the authors have proposed techniques to deal with α-equivalence modulo the equational theories A, C and AC using the standard approach via freshness constraints. The works [3, 2] show that despite the fact that C-unification problems have solutions generated by a finite family of fixed-point equations, there is no finitary representation of the admissible set of solutions using only freshness constraints and substitutions. Also, in [15] it is shown how nominal unification problems in a language with recursive let operators gives rise to solutions expressed in terms of freshness constraints and nominal fixed-point equations.

In this paper, we will develop an extension of fixed-point constraints modulo commutativity, namely, $\curlywedge_C$, and provide a set of rules for checking fixed-point judgements and α-equivalence judgements modulo C, which will provide a finitary representation of nominal C-unification solutions, consisting only of primitive fixed-point constraints and substitutions.

Overview

Section 2 presents the required preliminaries on nominal syntax. Section 3 introduces nominal α-equivalence using fixed-point constraints instead of freshness constraints. Section 4 introduces a sound and complete rule-based algorithm for nominal unification using fixed-point constraints. Before concluding, Section 5 shows how fixed-point constraints are used to finitely represent solutions of fixed-point equations, and so of nominal C-unification problems.

2 Preliminaries

We assume the reader is familiar with the notions of *nominal set* and *nominal syntax*. In this section we recall the main concepts and notations that are needed in this paper; for more details we refer the reader to [14, 16].

2.1 Nominal Terms

Let $\mathbb{A}$ be a fixed and countably infinite set of elements $a, b, c, \ldots$, which will be called *atoms* (atomic names). A permutation on $\mathbb{A}$ is a bijection on $\mathbb{A}$ with finite domain.

Fix a countably infinite set $\mathcal{X} = \{X, Y, Z, \ldots\}$ of variables and a countable set $\mathcal{F} = \{\mathsf{f}, \mathsf{g}, \ldots\}$ of function symbols.

▶ **Definition 1** (Nominal grammar). Nominal terms are generated by the following grammar.

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

where a is an *atom term*, $[a]t$ denotes the *abstraction* of the atom a over the term t, $(t_1, \ldots, t_n)$ is a tuple, $\mathsf{f}\ t$ denotes the *application of* f *to* t and $\pi \cdot X$ is a *moderated variable* or *suspension*, where π is an atom permutation.

We follow the *permutative convention* [11, Convention 2.3] for atoms throughout the paper, i.e., atoms a, b, c range permutatively over $\mathbb{A}$ so that they are always pairwise different, unless stated otherwise.

Atom *permutations* are represented by finite lists of *swappings*, which are pairs of different atoms $(a\ b)$; hence, a permutation π is generated by the following grammar:

$$\pi := Id \mid (a\ b)\pi.$$

We call Id the identity permutation, which is usually omitted from the list of swappings defining a permutation. Suspensions of the form $Id \cdot X$ will be represented just by X. We write π^{-1} for the *inverse* of π, and use $\circ$ to denote the composition of permutations. For example, if $\pi = (a\ b)(b\ c)$ then $\pi(c) = a$ and $c = \pi^{-1}(a)$.

The *difference set* of two permutations π, π' is $\mathtt{ds}(\pi, \pi') = \{a \mid \pi(a) \neq \pi'(a)\}$.

We write $\mathtt{Var}(t)$ for the set of variables occurring in t. Ground terms are terms without variables, that is $\mathtt{Var}(t) = \emptyset$. A ground term may still contain atoms, for example a is a ground term and X is not.

▶ **Definition 2** (Permutation action). The action of a permutation π on a term t is defined by induction on the number of swappings in π:

$Id \cdot t = t$ and $((a\ b)\pi) \cdot t = (a\ b) \cdot (\pi \cdot t)$, where

$$\begin{array}{l}(a\ b) \cdot a = b,\\ (a\ b) \cdot b = a,\\ (a\ b) \cdot c = c\end{array} \qquad \begin{array}{ll}(a\ b) \cdot (\pi \cdot X) = ((a\ b) \circ \pi) \cdot X, & (a\ b) \cdot [c]t = [(a\ b) \cdot c](a\ b) \cdot t\\ (a\ b) \cdot \mathsf{f}\ t = \mathsf{f}\ (a\ b) \cdot t, & (a\ b) \cdot (t_1, \ldots, t_n) = ((a\ b) \cdot t_1, \ldots, (a\ b) \cdot t_n)\end{array}$$

▶ **Definition 3** (Substitution). *Substitutions* are generated by the grammar

$$\sigma ::= id \mid [X \mapsto s]\sigma.$$

Postfix notation is used for substitution application and $\circ$ for composition: $t(\sigma \circ \sigma') = (t\sigma)\sigma'$. Substitutions act on terms elementwise in the natural way: $t\ id = t$, $t[X \mapsto s]\sigma = (t[X \mapsto s])\sigma$, where

$$\begin{array}{rl}a[X \mapsto s] = a & (t_1, \ldots, t_n)[X \mapsto s] = (t_1[X \mapsto s], \ldots, t_n[X \mapsto s])\\ (\mathsf{f}\ t)[X \mapsto s] = \mathsf{f}(t[X \mapsto s]) & (\pi \cdot X)[X \mapsto s] = \pi \cdot s\\ ([a]t)[X \mapsto s] = [a](t[X \mapsto s]) & (\pi \cdot Y)[X \mapsto s] = \pi \cdot Y\end{array}$$

2.2 Nominal sets and support

Let S be a set equipped with an action of the group $\texttt{Perm}(\mathbb{A})$ of finite permutations of $\mathbb{A}$.

▶ **Definition 4.** A set $A \subset \mathbb{A}$ is a *support* for an element $x \in S$ if for all $\pi \in \texttt{Perm}(\mathbb{A})$, the following holds

$$((\forall a \in A)\ \pi(a) = a) \Rightarrow \pi \cdot x = x \tag{1}$$

A *nominal set* is a set equipped with an action of the group $\texttt{Perm}(\mathbb{A})$, that is, a $\texttt{Perm}(\mathbb{A})$-set, all of whose elements have finite support.

As in [14], we denote by $\texttt{supp}_S(x)$ the least finite support of x, that is,

$$\texttt{supp}_S(x) := \bigcap \{A \in \mathcal{P}(\mathbb{A}) \mid A \text{ is a finite support for } x\}.$$

We write $\texttt{supp}(x)$ when S is clear from the context. Clearly, each $a \in \mathbb{A}$ is finitely supported by $\{a\}$, therefore $\texttt{supp}(a) = \{a\}$.

3 Constraints

The native notion of equality on nominal terms is α-equivalence, written $s \approx_\alpha t$. This relation is usually axiomatised using a *freshness relation* between atoms and terms, written $a\#t$ – read "a fresh for t", which, intuitively, corresponds to the idea of an atom not occurring free in a term (see for instance [16, 8]). However, freshness is not a primitive notion in nominal sets; it is derived using the quantifier И combined with a notion of fixed-point, as shown by Pitts [14]:

$$a\#X \Leftrightarrow И a'.(a\ a') \cdot X = X.$$

In this work, instead of defining α-equivalence using freshness, we define it using the more primitive notion of *fixed-point* under the action of permutations. We will denote this relation $\overset{\curlywedge}{\approx}_\alpha$, and show that it coincides with $\approx_\alpha$ on ground terms, i.e., the relation defined using fixed-points of permutations corresponds to the relation defined using freshness. For non-ground terms, there is also a correspondence, but under different kinds of assumptions (fixed-point constraints vs. freshness constraints).

3.1 Fixed-points of permutations and term equality

We start by defining a binary relation that describes which elements of a nominal set S are fixed-points of a permutation $\pi \in \texttt{Perm}(\mathbb{A})$:

▶ **Definition 5** (Fixed-point relation)**.** Let S be a nominal set. The *fixed-point relation* $\curlywedge \subseteq \texttt{Perm}(\mathbb{A}) \times S$ is defined as: $\pi \curlywedge x \Leftrightarrow dom(\pi) \cap \texttt{supp}(x) = \emptyset$. Read "$\pi \curlywedge x$" as "*$\pi$ fixes x*".

The fixed-point relation between permutations and terms will play an important role in the definition of α-equality. Below we define the *fixed-point* constraints and *equality* constraints using predicates $\curlywedge$ and $\overset{\curlywedge}{\approx}_\alpha$ and then give deduction rules to derive fixed-point and equality judgements. Intuitively,

- $s \overset{\curlywedge}{\approx}_\alpha t$ will mean that s and t are α-equivalent, i.e., equivalent modulo renaming of abstracted atoms.

$$\frac{\pi(a) = a}{\Upsilon \vdash \pi \curlywedge a}\ (\curlywedge\mathbf{a}) \qquad \frac{\mathtt{supp}(\pi^{\pi'^{-1}}) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash \pi \curlywedge \pi' \cdot X}\ (\curlywedge\mathbf{var})$$

$$\frac{\Upsilon \vdash \pi \curlywedge t}{\Upsilon \vdash \pi \curlywedge \mathsf{f}\ t}\ (\curlywedge\mathsf{f}) \qquad \frac{\Upsilon \vdash \pi \curlywedge t_1 \quad \ldots \quad \Upsilon \vdash \pi \curlywedge t_n}{\Upsilon \vdash \pi \curlywedge (t_1, \ldots, t_n)}\ (\curlywedge\mathbf{tuple})$$

$$\frac{\Upsilon, (c_1\ c_2) \curlywedge \mathtt{Var}(t) \vdash \pi \curlywedge (a\ c_1) \cdot t}{\Upsilon \vdash \pi \curlywedge [a]t}\ (\curlywedge\mathbf{abs}),\ \begin{array}{l} c_1 \text{ and } c_2 \\ \text{new names} \end{array}$$

Figure 1 Fixed-point rules.

- $\pi \curlywedge t$ will mean that the permutation π fixes the nominal term t, that is, $\pi \cdot t \overset{\curlywedge}{\approx}_\alpha t$. This means that π has "no effect" on t except for the renaming of bound names, for instance, $(a\ b) \curlywedge [a]a$ but not $(a\ b) \curlywedge \mathsf{f}\ a$.

▶ **Definition 6** (Fixed-point and equality constraints)**.** A *fixed-point constraint* is a pair $\pi \curlywedge t$ of a permutation π and a term t. An *α-equivalence constraint* is a pair of the form $s \overset{\curlywedge}{\approx}_\alpha t$. We call a fixed-point constraint of the form $\pi \curlywedge X$ a *primitive fixed-point constraint* and a set of such constraints is called a *fixed-point context*. $\Upsilon, \Psi, \ldots$ range over fixed-point contexts. We write $\pi \curlywedge \mathtt{Var}(t)$ as an abbreviation for the set of constraints $\{\pi \curlywedge X \mid X \in \mathtt{Var}(t)\}$.

The set of variables $\mathtt{Var}(\Upsilon)$ is defined as expected. The set of permutations of a fixed-point context Υ with respect to the variable $X \in \mathtt{Var}(\Upsilon)$, denoted by $\mathtt{perm}(\Upsilon|_X)$, is defined as $\mathtt{perm}(\Upsilon|_X) := \{\pi \mid \pi \curlywedge X \in \Upsilon\}$. For a substitution σ and a fixed-point context Υ we define $\Upsilon\sigma := \{\pi \curlywedge X\sigma \,|\, \pi \curlywedge X \in \Upsilon\}$.

To define the relation $\curlywedge$, we rely on the notion of *conjugation* of permutations. The conjugate of π with respect to ρ, denoted as π^ρ, is the result of the composition: $\rho \circ \pi \circ \rho^{-1}$.

$$\begin{array}{rccccccc} \pi^\rho : & A & \overset{\rho^{-1}}{\to} & A & \overset{\pi}{\to} & A & \overset{\rho}{\to} & A \\ & a & \mapsto & \rho^{-1}(a) & \mapsto & \pi(\rho^{-1}(a)) & \mapsto & \rho(\pi(\rho^{-1}(a))) \end{array}$$

▶ **Definition 7** (Judgements)**.** A *fixed-point judgement* is a tuple $\Upsilon \vdash \pi \curlywedge t$ of a fixed-point context and a fixed-point constraint. An *α-equivalence judgement* is a tuple $\Psi \vdash s \overset{\curlywedge}{\approx}_\alpha t$ of a fixed-point context and an equality constraint. The derivable fixed-point and α-equivalence judgements are defined by the rules in Figures 1 and 2.

▶ **Example 8.** The term $[a]\mathsf{f}a$ is a fixed-point for the permutation $(a\ b)$, since $(a\ b)[a]\mathsf{f}a \approx_\alpha [b]\mathsf{f}b$, therefore, $(a\ b) \curlywedge [a]\mathsf{f}a$. However, $\mathsf{f}a$ is not a fixed-point for $(a\ b)$, since $(a\ b) \cdot \mathsf{f}a \not\approx_\alpha \mathsf{f}b$.

Rule $(\curlywedge\mathbf{a})$ states that if $a \notin dom(\pi)$, then a is a fixed-point of π.

In rule $(\curlywedge\mathbf{var})$, the condition $\mathtt{supp}(\pi^{\pi'^{-1}}) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ means that the permutation can be generated from $\mathtt{perm}(\Upsilon|_X)$, hence it fixes X. Rules $(\curlywedge\mathsf{f})$ and $(\curlywedge\mathtt{tuple})$ are straightforward. Rule $(\curlywedge\mathbf{abs})$ is the most interesting one. The intuition behind this rule is the following: $[a]t$ is a fixed-point of π if $\pi \cdot [a]t$ is α-equivalent to $[a]t$, that is, $[\pi(a)]\pi \cdot t$ is α-equivalent to $[a]t$; the latter means that the only atom that could be affected by π is a, hence, if we replace occurrences of a in t with another, new atom c_1, π should have no effect.

The α-equality relation is defined in terms of fixed-point constraints. Rules $(\overset{\curlywedge}{\approx}_\alpha\ \mathsf{a})$, $(\overset{\curlywedge}{\approx}_\alpha\ \mathsf{f})$, $(\overset{\curlywedge}{\approx}_\alpha\ [\mathsf{a}])$ and $(\overset{\curlywedge}{\approx}_\alpha\ \mathsf{tuple})$ are defined as expected, whereas the intuition behind rule $(\overset{\curlywedge}{\approx}_\alpha\ \mathsf{var})$ is similar to the corresponding rule in Figure 1. The most interesting rule is $(\overset{\curlywedge}{\approx}_\alpha\ \mathsf{ab})$.

$$\frac{}{\Upsilon \vdash a \mathrel{\dot{\approx}}_\alpha a}\,(\dot{\approx}_\alpha \mathbf{a}) \qquad \frac{\mathtt{supp}((\pi')^{-1} \circ \pi) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash \pi \cdot X \mathrel{\dot{\approx}}_\alpha \pi' \cdot X}\,(\dot{\approx}_\alpha \mathbf{var})$$

$$\frac{\Upsilon \vdash t \mathrel{\dot{\approx}}_\alpha t'}{\Upsilon \vdash \mathtt{f}\ t \mathrel{\dot{\approx}}_\alpha \mathtt{f}\ t'}\,(\dot{\approx}_\alpha \mathtt{f}) \qquad \frac{\Upsilon \vdash t_1 \mathrel{\dot{\approx}}_\alpha t'_1 \quad \ldots \quad \Upsilon \vdash t_n \mathrel{\dot{\approx}}_\alpha t'_n}{\Upsilon \vdash (t_1, \ldots, t_n) \mathrel{\dot{\approx}}_\alpha (t'_1, \ldots, t'_n)}\,(\dot{\approx}_\alpha \mathbf{tuple})$$

$$\frac{\Upsilon \vdash t \mathrel{\dot{\approx}}_\alpha t'}{\Upsilon \vdash [a]t \mathrel{\dot{\approx}}_\alpha [a]t'}\,(\dot{\approx}_\alpha [\mathbf{a}]) \qquad \frac{\Upsilon \vdash s \mathrel{\dot{\approx}}_\alpha (a\ b) \cdot t \quad \Upsilon, (c_1\ c_2) \curlywedge \mathtt{Var}(t) \vdash (a\ c_1) \curlywedge t}{\Upsilon \vdash [a]s \mathrel{\dot{\approx}}_\alpha [b]t}\,(\dot{\approx}_\alpha \mathbf{ab})$$

Figure 2 Rules for equality. In rule $(\dot{\approx}_\alpha \mathbf{ab})$, c_1 and c_2 are new names.

Intuitively, it states that for two abstractions $[a]s$ and $[b]t$ to be equivalent, we must obtain equivalent terms if we rename in one of them, in our case t, the abstracted atom b to a, so that they both use the same atom. Moreover, the atom a should not occur free in t, which is checked by stating that $(a\ c_1)$ fixes t for some new atom c_1 that is not in the support of the variables occurring in t.

We prove below that $\dot{\approx}_\alpha$ is indeed an equivalence relation, for which we need to study the properties of the relations $\dot{\approx}_\alpha$ and $\curlywedge$, starting with *inversion* and *equivariance*.

▶ **Lemma 9** (Inversion). *The inference rules for $\dot{\approx}_\alpha$ are invertible.*

The notion of *equivariance* relies on the conjugation of the permutation π by ρ, π^ρ. The following basic property is used in the proofs in this section.

▶ **Lemma 10.** *Let ρ be a permutation in $\mathtt{Perm}(\mathbb{A})$ and a, b atoms in $\mathbb{A}$. Then $(a\ b)^\rho = (\rho(a)\ \rho(b))$.*

▶ **Lemma 11.**

i.) *$\Upsilon \vdash \pi \curlywedge t$ if and only if $\mathtt{supp}(\pi) \cap \mathtt{supp}(t) = \emptyset$.*

ii.) *If $\Upsilon \vdash s \mathrel{\dot{\approx}}_\alpha t$ then $\mathtt{supp}(s) = \mathtt{supp}(t)$.*

Proof. Both parts are proved by induction. In part (i), we analyse cases depending on the last rule applied in the derivation of $\Upsilon \vdash \pi \curlywedge t$. We show the cases for rules $(\curlywedge\mathbf{var})$ and $(\curlywedge\mathbf{abs})$, the other cases follow directly by induction.

If the last rule applied is $(\curlywedge\mathbf{var})$ then $t = \pi' \cdot X$ and $\Upsilon \vdash \pi \curlywedge \pi' \cdot X$ if and only if (Inversion Lemma) $\mathtt{supp}(\pi^{\pi'^{-1}}) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$, if and only if $\mathtt{supp}(\pi) \subseteq \pi' \cdot \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$. Since $\mathtt{supp}(X) \cap \mathtt{supp}(\mathtt{perm}(\Upsilon|_X)) = \emptyset$ by Definition 5, we deduce $\mathtt{supp}(\pi) \cap \mathtt{supp}(\pi' \cdot X) = \emptyset$ as required.

If the last rule applied is $(\curlywedge\mathbf{abs})$, then $t = [a]t'$ and $\Upsilon \vdash \pi \curlywedge [a]t'$ if and only if (Inversion Lemma) $\Upsilon, (c_1\ c_2) \curlywedge \mathtt{Var}(t') \vdash \pi \curlywedge (a\ c_1) \cdot t'$. By induction, $\mathtt{supp}(\pi) \cap \mathtt{supp}((a\ c_1)t') = \emptyset$ and since $\mathtt{supp}([a]t') = \mathtt{supp}((a\ c_1) \cdot t') - \{c_1\}$ (because c_1 is a new atom and $(c_1\ c_2) \curlywedge \mathtt{Var}(t')$), we obtain $\mathtt{supp}(\pi) \cap \mathtt{supp}([a]t') = \emptyset$ as required.

The proof for part (ii), by induction on the derivation of $\Upsilon \vdash s \mathrel{\dot{\approx}}_\alpha t$, is similar. In the case of rule $(\dot{\approx}_\alpha\ \mathbf{var})$, the premise implies that $\mathtt{ds}(\pi, \pi') \cap \mathtt{supp}(X) = \emptyset$, hence $\mathtt{supp}(\pi \cdot X) = \mathtt{supp}(\pi' \cdot X)$. In the case of rule $(\dot{\approx}_\alpha\ \mathbf{ab})$, by induction hypothesis $\mathtt{supp}(s) = \mathtt{supp}((a\ b) \cdot t)$ and since we know that $(a\ c_1) \curlywedge t$, using part 1 we obtain the result. ◀

▶ **Lemma 12** (Equivariance).
i.) $\Upsilon \vdash \pi \between t$ *iff* $\Upsilon \vdash \pi^{\rho} \between \rho \cdot t$, *for any permutation* ρ.
ii.) *If* $\Upsilon \vdash s \mathrel{\overset{\between}{\approx}_{\alpha}} t$ *then* $\Upsilon \vdash \pi \cdot s \mathrel{\overset{\between}{\approx}_{\alpha}} \pi \cdot t$.

Proof. By induction on the rules of Figures 1 and 2. ◀

▶ **Lemma 13** ($\between$ preservation under $\mathrel{\overset{\between}{\approx}_{\alpha}}$). *If* $\Upsilon \vdash s \mathrel{\overset{\between}{\approx}_{\alpha}} t$ *and* $\Upsilon \vdash \pi \between s$ *then* $\Upsilon \vdash \pi \between t$.

Proof. Direct consequence of Lemma 11. ◀

▶ **Proposition 14** (Strengthening for $\between$). *If* $\Upsilon, \pi \between X \vdash \pi' \between s$ *and* $\mathtt{supp}(\pi) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ *or* $X \notin \mathtt{Var}(s)$ *then* $\Upsilon \vdash \pi' \between s$.

▶ **Proposition 15** (Strengthening for $\mathrel{\overset{\between}{\approx}_{\alpha}}$). *If* $\Upsilon, \pi \between X \vdash s \mathrel{\overset{\between}{\approx}_{\alpha}} t$ *and* $\mathtt{supp}(\pi) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ *or* $X \notin \mathtt{Var}(s,t)$, *then* $\Upsilon \vdash s \mathrel{\overset{\between}{\approx}_{\alpha}} t$.

▶ **Proposition 16** (Weakening). *Suppose that* $\Upsilon \vdash \Upsilon'\sigma$. *Then,*
1. $\Upsilon' \vdash \pi \between s \Longrightarrow \Upsilon \vdash \pi \between s\sigma$.
2. $\Upsilon' \vdash s \mathrel{\overset{\between}{\approx}_{\alpha}} t \Longrightarrow \Upsilon \vdash s\sigma \mathrel{\overset{\between}{\approx}_{\alpha}} t\sigma$.

Proof. By induction on the rules of Figures 1 and 2. ◀

▶ **Example 17.** Notice that $(a\ c) \between X \vdash (a\ b) \between (b\ c) \cdot X$, for

$$(a\ c) \between X \vdash (a\ b)^{(b\ c)} \between X \Leftrightarrow (a\ c) \between X \vdash (a\ c) \between X \quad \textit{(by Equivariance)} \tag{2}$$

The following correctness property states that $\between$ is indeed the fixed-point relation:

▶ **Theorem 18.** *Let* Υ, π *and* t *be a fixed-point context, a permutation and a nominal term, respectively.* $\Upsilon \vdash \pi \between t$ *iff* $\Upsilon \vdash \pi \cdot t \mathrel{\overset{\between}{\approx}_{\alpha}} t$.

Sketch. In both directions the proof follows by induction on the structure of the term t and by case analysis on the last rule applied in the derivation. We show only $\Upsilon \vdash \pi \between t \Rightarrow \Upsilon \vdash \pi \cdot t \mathrel{\overset{\between}{\approx}_{\alpha}} t$. Below we sketch the interesting cases, the other cases follow by induction hypothesis easily.
1. The last rule is ($\between$**var**). In this case, $t = \pi' \cdot X$ and $\mathtt{supp}((\pi')^{-1} \circ \pi \circ \pi') \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ and therefore, $\pi \cdot (\pi' \cdot X) \mathrel{\overset{\between}{\approx}_{\alpha}} \pi' \cdot X$, via rule ($\mathrel{\overset{\between}{\approx}_{\alpha}}$ **var**).
2. The last rule is ($\between$**abs**). In this case, $t = [a]t'$ and $\pi \between t$ has a derivation of the form:
$$\dfrac{\dfrac{\Pi}{\Upsilon, (c_1\ c_2) \between \mathtt{Var}(t') \vdash \pi \between (a\ c_1) \cdot t'}}{\Upsilon \vdash \pi \between [a]t'}$$
From $\Upsilon, (c_1\ c_2) \between \mathtt{Var}(t') \vdash \pi \between (a\ c_1) \cdot t'$ it follows, from Lemma 11:
$$\mathtt{supp}(\pi) \cap \mathtt{supp}((a\ c_1) \cdot t') = \emptyset. \tag{3}$$
We need to prove that $\Upsilon \vdash [\pi(a)]\pi \cdot t' \mathrel{\overset{\between}{\approx}_{\alpha}} [a]t'$, that is, $\Upsilon \vdash \pi \cdot t' \mathrel{\overset{\between}{\approx}_{\alpha}} (\pi(a)\ a) \cdot t'$ and also $\Upsilon, (c_1\ c_2) \between \mathtt{Var}(t') \vdash (\pi(a)\ c_1) \between t'$ for some new atoms c_1, c_2.
By IH, there exist a proof Π' for $\Upsilon, (c_1\ c_2) \between \mathtt{Var}(t') \vdash \pi \cdot ((a\ c_1).t') \mathrel{\overset{\between}{\approx}_{\alpha}} (a\ c_1) \cdot t'$. Let $\Upsilon' = \Upsilon, (c_1\ c_2) \between \mathtt{Var}(t')$. The following equivalence holds:
$$\Upsilon' \vdash \pi.((a\ c_1) \cdot t') \mathrel{\overset{\between}{\approx}_{\alpha}} (a\ c_1) \cdot t' \iff \Upsilon' \vdash (\pi(a)\ c_1) \cdot (\pi \cdot t') \mathrel{\overset{\between}{\approx}_{\alpha}} (a\ c_1) \cdot t' \tag{4}$$
Also, $\Upsilon' \vdash (\pi \cdot t') \mathrel{\overset{\between}{\approx}_{\alpha}} (\pi(a)\ c_1) \cdot ((a\ c_1) \cdot t')$ by Equivariance. And since $\Upsilon' \vdash (\pi(a)\ c_1) \cdot ((a\ c_1) \cdot t') \mathrel{\overset{\between}{\approx}_{\alpha}} (\pi(a)\ a) \cdot t'$, we are done. ◀

$$\frac{}{\Delta \vdash a\#b}\,(\#\mathbf{a}) \qquad \frac{\pi^{-1}(a)\#X \in \Delta}{\Delta \vdash a\#\pi\cdot X}\,(\#\mathbf{var})$$

$$\frac{\Delta \vdash a\#t}{\Delta \vdash a\#\mathtt{f}\ t}\,(\#\mathtt{f}) \qquad \frac{\Delta \vdash a\#t_1 \quad \ldots \quad \Delta \vdash a\#t_n}{\Delta \vdash a\#(t_1,\ldots,t_n)}\,(\#\mathbf{tuple})$$

$$\frac{}{\Delta \vdash a\#[a]t}\,(\#[\mathbf{a}]) \qquad \frac{\Delta \vdash a\#t}{\Upsilon \vdash a\#[b]t}\,(\#\mathbf{abs})$$

Figure 3 Rules for freshness.

$$\frac{}{\Delta \vdash a \approx_\alpha a}\,(\approx_\alpha \mathbf{a}) \qquad \frac{\mathtt{ds}(\pi,\pi')\#X \subseteq \Delta}{\Delta \vdash \pi\cdot X \approx_\alpha \pi'\cdot X}\,(\approx_\alpha \mathbf{var})$$

$$\frac{\Delta \vdash t \approx_\alpha t'}{\Delta \vdash \mathtt{f}\ t \approx_\alpha \mathtt{f}\ t'}\,(\approx_\alpha \mathtt{f}) \qquad \frac{\Delta \vdash t_1 \approx_\alpha t_1' \quad \ldots \quad \Delta \vdash t_n \approx_\alpha t_n'}{\Delta \vdash (t_1,\ldots,t_n) \approx_\alpha (t_1',\ldots,t_n')}\,(\approx_\alpha \mathbf{tuple})$$

$$\frac{\Delta \vdash t \approx_\alpha t'}{\Delta \vdash [a]t \approx_\alpha [a]t'}\,(\approx_\alpha [\mathbf{a}]) \qquad \frac{\Delta \vdash\ s \approx_\alpha (a\ b).t \quad \Delta \vdash a\#t}{\Upsilon \vdash [a]s \approx_\alpha [b]t}\,(\approx_\alpha \mathbf{ab})$$

Figure 4 Rules for α-equality via freshness.

3.2 From freshness to fixed-point constraints

In this section we show that the α-equivalence relation defined in terms of *freshness constraints*, denoted as $\approx_\alpha$, is equivalent to $\overset{\curlywedge}{\approx}_\alpha$, given that a transformation $[\_]^{\curlywedge}$ from freshness to fixed-point constraints and a transformation $[\_]^{\#}$ from fixed-point to freshness constraints can be defined. In the standard approach [13, 8], the freshness relation ($a\#t$) and the α-equivalence relation $s \approx_\alpha t$ (w.r.t. $\#$), are axiomatised using the rules in Figures 3 and 4, respectively.

To define $\approx_\alpha$ we use the *difference set* of two permutations in rule ($\approx_\alpha$ **var**), and $\mathtt{ds}(\pi,\pi')\#X = \{a\#X \mid a \in \mathtt{ds}(\pi,\pi')\}$.

The symbols Δ and ∇ denote *freshness contexts*, that is, sets of freshness constraints of the form $a\#X$, meaning that a is fresh in X. The domain of a freshness context Δ, denoted by $\mathtt{dom}(\Delta)$, consists of the atoms occurring in Δ; $\Delta|_X$ consists of the restriction of Δ to the freshness constraints on variable X, that is, the set $\{a\#X \mid a\#X \in \Delta\}$. Below we denote by $\mathfrak{F}_\#$ the family of freshness contexts, and by $\mathfrak{F}_{\curlywedge}$ the family of fixed-point contexts. The mapping $[\_]_{\curlywedge}$ below associates each freshness constraint in Δ with a fixed-point constraint:

$$\begin{array}{rccl} [\_]_{\curlywedge}: & \Delta & \longrightarrow & \mathfrak{F}_{\curlywedge} \\ & a\#X & \mapsto & (a\ c_a) \curlywedge X \text{ where } c_a \text{ is a new name.} \end{array}$$

We denote by $[\Delta]_{\curlywedge}$ the image of Δ under $[\_]_{\curlywedge}$.

The mapping $[\_]_{\curlywedge}$ below associates each fixed-point constraint in Υ with a freshness constraint:

$$\begin{array}{rccl} [\_]_{\#}: & \Upsilon & \longrightarrow & \mathfrak{F}_{\#} \\ & \pi \curlywedge X & \mapsto & \mathtt{supp}(\pi)\#X. \end{array}$$

We denote by $[\Upsilon]_{\#}$ the image of Υ under $[\_]_{\#}$.

▶ **Lemma 19.**
1. $\Delta \vdash a\#t \Leftrightarrow [\Delta]_{\curlywedge}, \overline{(c_2\ c_1)} \curlywedge \mathtt{Var}(t) \vdash (a\ c_1) \curlywedge t$.
2. $\Upsilon \vdash \pi \curlywedge t \Leftrightarrow [\Upsilon]_{\#} \vdash \mathtt{supp}(\pi)\#t$.

▶ **Theorem 20.** $\overset{\curlywedge}{\approx}_\alpha$ *coincides with* $\approx_\alpha$ *on ground terms, that is,* $\vdash s \approx_\alpha t \iff \vdash s \overset{\curlywedge}{\approx}_\alpha t$. *More generally,*
1. $\Delta \vdash s \approx_\alpha t \Rightarrow [\Delta]_{\curlywedge} \vdash s \overset{\curlywedge}{\approx}_\alpha t$.
2. $\Upsilon \vdash s \overset{\curlywedge}{\approx}_\alpha t \Rightarrow [\Upsilon]_{\#} \vdash s \approx_\alpha t$.

Sketch.
1. The proof is by induction on the derivation of $\Delta \vdash s \approx_\alpha t$. The interesting case is when the derivation is an instance of ($\approx_\alpha$ **var**):
$$\frac{\mathtt{ds}(\pi, \pi_1)\#X \subseteq \Delta}{\Delta \vdash \pi \cdot X \approx_\alpha \pi_1 \cdot X}\ (\approx_\alpha \textbf{var})$$
We want to show that $[\Delta]_{\curlywedge} \vdash \pi \cdot X \overset{\curlywedge}{\approx}_\alpha \pi_1 \cdot X$. To use rule ($\overset{\curlywedge}{\approx}_\alpha$ **var**), we need to show that $\mathtt{supp}(\pi_1^{-1} \circ \pi) \subseteq \mathtt{supp}(\mathtt{perm}(([\Delta]_{\curlywedge})|_X))$. Let $b \in \mathtt{supp}(\pi_1^{-1} \circ \pi)$ and suppose $b \notin \mathtt{ds}(\pi, \pi_1)$. Then $\pi(b) = \pi_1(b)$ and $\pi_1^{-1}(\pi(b)) = b$, contradiction. Therefore, $b \in \mathtt{ds}(\pi, \pi_1)$ and $(b\ c_b) \curlywedge X \in [\Delta]_{\curlywedge}$ (for c_b a new name), and the result follows. ◀

As a corollary, since $\approx_\alpha$ is an equivalence relation [16], we deduce that $\overset{\curlywedge}{\approx}_\alpha$ is also an equivalence relation.

▶ **Theorem 21.** $\overset{\curlywedge}{\approx}_\alpha$ *is an equivalence relation.*

4 Nominal Unification via fixed-point constraints

In this section we define the notion of nominal unification in terms of fixed-point constraints.

▶ **Definition 22.** A *unification problem* $\mathtt{Pr}$ consists of a finite set of equations and fixed-point constraints of the form $s \overset{\curlywedge^?}{\approx}_\alpha t$ and $\pi \curlywedge^? t$, respectively.

We design a unification algorithm via the simplification rules presented in Table 1. These rules act on unification problems $\mathtt{Pr}$. We abbreviate $(t_1, \ldots, t_n)$ as $(\widetilde{t})_n$, and for a set S, $\overline{\pi \curlywedge S} = \{\pi \curlywedge X \mid X \in S\}$.

We write $\mathtt{Pr} \Longrightarrow \mathtt{Pr}'$, when $\mathtt{Pr}'$ is obtained from $\mathtt{Pr}$ by applying a simplification rule from Table 1 and we write $\overset{*}{\Longrightarrow}$ for the reflexive and transitive closure of $\Longrightarrow$.

▶ **Lemma 23.** *There is no infinite chain of reductions* $\Longrightarrow$ *starting from a problem* $\mathtt{Pr}$.

Proof. Termination of the simplification rules follows directly from the fact that the following measure of the size of $\mathtt{Pr}$ is strictly decreasing:
$[\mathtt{Pr}] = (n_1, M)$ where n_1 is the number of different variables used in $\mathtt{Pr}$, and M is the multiset of sizes of equality constraints and non-primitive fixed-point constraints occurring in $\mathtt{Pr}$.

Each simplification step either eliminates one variable (when an instantiation rule is used) and therefore decreases the first component of the interpretation, or leaves the first component unchanged but replaces a constraint with smaller ones and/or primitive ones. ◀

The normal form of $\mathtt{Pr}$ by $\Longrightarrow$ is defined as expected and denoted by $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$.

We say that an equality constraint $s \overset{\curlywedge^?}{\approx}_\alpha t$ is *reduced* when one of the following holds:
1. $s := a$ and $t := b$ are distinct atoms;
2. s and t are headed with different function symbols, that is, $s := \mathtt{f}\ s'$ and $t := \mathtt{g}\ t'$;

Table 1 Simplification Rules for Problems. In ($\curlywedge abs$) and ($\overset{\curlywedge}{\approx}_\alpha abs2$), c_1 and c_2 are new names.

($\curlywedge at$)	$\mathtt{Pr} \uplus \{\pi \curlywedge^? a\}$	$\Longrightarrow$	$\mathtt{Pr}$, if $\pi(a) = a$
($\curlywedge f$)	$\mathtt{Pr} \uplus \{\pi \curlywedge^? \mathsf{f}t\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge^? t\}$
($\curlywedge t$)	$\mathtt{Pr} \uplus \{\pi \curlywedge^? (\widetilde{t})_n\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge^? t_1, \ldots, \pi \curlywedge^? t_n\}$
($\curlywedge abs$)	$\mathtt{Pr} \uplus \{\pi \curlywedge^? [a]t\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge^? (a\ c_1) \cdot t, \overline{(c_1\ c_2) \curlywedge^? \mathtt{Var}(t)}\}$
($\curlywedge var$)	$\mathtt{Pr} \uplus \{\pi \curlywedge^? \pi' \cdot X\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi^{(\pi')^{-1}} \curlywedge^? X\}$, if $\pi' \neq Id$
($\overset{\curlywedge}{\approx}_\alpha a$)	$\mathtt{Pr} \uplus \{a \overset{\curlywedge^?}{\approx}_\alpha a\}$	$\Longrightarrow$	$\mathtt{Pr}$
($\overset{\curlywedge}{\approx}_\alpha f$)	$\mathtt{Pr} \uplus \{\mathsf{f}\ t \overset{\curlywedge^?}{\approx}_\alpha \mathsf{f}\ t'\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \approx_\alpha{}^? t'\}$
($\overset{\curlywedge}{\approx}_\alpha t$)	$\mathtt{Pr} \uplus \{(\widetilde{t})_n \approx_\alpha{}^? (\widetilde{t'})_n\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t_1 \overset{\curlywedge^?}{\approx}_\alpha t'_1, \ldots, t_n \overset{\curlywedge^?}{\approx}_\alpha t'_n\}$
($\overset{\curlywedge}{\approx}_\alpha abs1$)	$\mathtt{Pr} \uplus \{[a]t \overset{\curlywedge^?}{\approx}_\alpha [a]t'\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \overset{\curlywedge^?}{\approx}_\alpha t'\}$
($\overset{\curlywedge}{\approx}_\alpha abs2$)	$\mathtt{Pr} \uplus \{[a]t \overset{\curlywedge^?}{\approx}_\alpha [b]s\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \overset{\curlywedge^?}{\approx}_\alpha (a\ b) \cdot s, (a\ c_1) \curlywedge^? s, \overline{(c_1\ c_2) \curlywedge^? \mathtt{Var}(s)}\}$
($\approx_\alpha var$)	$\mathtt{Pr} \uplus \{\pi \cdot X \overset{\curlywedge^?}{\approx}_\alpha \pi' \cdot X\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{(\pi')^{-1} \circ \pi \curlywedge^? X\}$
($\approx_\alpha inst1$)	$\mathtt{Pr} \uplus \{\pi \cdot X \overset{\curlywedge^?}{\approx}_\alpha t\}$	$\overset{[X \mapsto \pi^{-1}.t]}{\Longrightarrow}$	$\mathtt{Pr}\{X \mapsto \pi^{-1}.t\}$, if $X \notin \mathtt{Var}(t)$
($\approx_\alpha inst2$)	$\mathtt{Pr} \uplus \{t \overset{\curlywedge^?}{\approx}_\alpha \pi \cdot X\}$	$\overset{[X \mapsto \pi^{-1}.t]}{\Longrightarrow}$	$\mathtt{Pr}\{X \mapsto \pi^{-1}.t\}$, if $X \notin \mathtt{Var}(t)$

3. s and t have different term constructors, that is, $s = [a]s'$ and $t = \mathsf{f}\ t'$, for some term former f, or $s = \pi \cdot X$ and $t = a$, etc.

A fixed-point constraint $\pi \curlywedge^? s$ is *reduced* when it is of the form $\pi \curlywedge^? a$ and $\pi(a) \neq a$, or $\pi \curlywedge^? X$, the former is called *inconsistent* whereas the latter is called *consistent*.

▶ **Example 24.** For $\mathtt{Pr} = [a]\mathsf{f}(X, a) \overset{\curlywedge^?}{\approx}_\alpha [b]\mathsf{f}((b\ c) \cdot W, (a\ c) \cdot Y)$, we obtain the following derivation chain:

$$[a]\mathsf{f}(X,a) \overset{\curlywedge^?}{\approx}_\alpha [b]\mathsf{f}((b\ c)\cdot W, (a\ c)\cdot Y) \Longrightarrow \left\{ \begin{array}{l} \mathsf{f}(X,a) \overset{\curlywedge^?}{\approx}_\alpha \mathsf{f}((a\ b)\circ(b\ c)\cdot W, (a\ b)\circ(a\ c)\cdot Y), \\ (a\ c_1) \curlywedge^? \mathsf{f}((b\ c)\cdot W, (a\ c)\cdot Y), \\ (c_2\ c_1) \curlywedge^? W, (c_2\ c_1) \curlywedge^? Y \end{array} \right\}$$

$$\Longrightarrow \left\{ \begin{array}{l} X \overset{\curlywedge^?}{\approx}_\alpha (a\ b)\circ(b\ c)\cdot W, a \overset{\curlywedge^?}{\approx}_\alpha (a\ b)\circ(a\ c)\cdot Y, \\ (a\ c_1) \curlywedge^? (b\ c)\cdot W, (a\ c_1) \curlywedge^? (a\ c)\cdot Y, (c_2\ c_1) \curlywedge^? W, (c_2\ c_1) \curlywedge^? Y \end{array} \right\}$$

$$\overset{[Y \mapsto b]}{\Longrightarrow} \left\{ X \overset{\curlywedge^?}{\approx}_\alpha (a\ b)\circ(b\ c)\cdot W, (a\ c_1)^{(b\ c)} \curlywedge^? W, (a\ c_1) \curlywedge^? b, (c_2\ c_1) \curlywedge^? W, (c_2\ c_1) \curlywedge^? b \right\}$$

$$\overset{*}{\Longrightarrow} \left\{ X \overset{\curlywedge^?}{\approx}_\alpha (a\ b)\circ(b\ c)\cdot W, (a\ c_1) \curlywedge^? W, (c_2\ c_1) \curlywedge^? W \right\}$$

$$\overset{[X \mapsto (a\ b)\circ(b\ c)\cdot W]}{\Longrightarrow} \left\{ (a\ c_1) \curlywedge^? W,\ (c_2\ c_1) \curlywedge^? W \right\} = \langle \mathtt{Pr} \rangle_{\mathtt{nf}}.$$

▶ **Definition 25.** Let $\mathtt{Pr}$ be a problem such that $\langle \mathtt{Pr} \rangle_{\mathtt{nf}} = \mathtt{Pr}'$. We say that $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$ is *reduced* when it consists of reduced constraints, and *successful* when $\mathtt{Pr}' = \emptyset$ or contains only consistent reduced fixed-point constraints; otherwise, $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$ *fails*.

▶ **Definition 26.** A *solution* for a problem $\mathtt{Pr}$ is a pair of the form $\langle \Phi, \sigma \rangle$ where the following conditions are satisfied:

1. $\Phi \vdash \pi \curlywedge t\sigma$, if $\pi \curlywedge^? t \in \mathtt{Pr}$;
2. $\Phi \vdash s\sigma \overset{\curlywedge^?}{\approx}_\alpha t\sigma$, if $s \overset{\curlywedge^?}{\approx}_\alpha t \in \mathtt{Pr}$.
3. $X\sigma = X\sigma\sigma$ for all $X \in \mathtt{Var}(\mathtt{Pr})$ (the substitution is idempotent).

The *solution set* for a problem $\mathtt{Pr}$ is denoted by $\mathcal{U}(\mathtt{Pr})$.

The simplification rules (Table 1) specify a *unification algorithm*: we apply the simplification rules in a problem $\mathtt{Pr}$ until we reach a normal form $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$. In the case $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$ *fails* or contains reduced equational constraints, we say that $\mathtt{Pr}$ is *unsolvable*; otherwise, $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$ is *solvable* and its solution consists of the composition σ of substitutions applied through the simplification steps and the fixed-point context $\Phi = \{\pi \curlywedge X \mid \pi \curlywedge^? X \in \langle \mathtt{Pr} \rangle_{\mathtt{nf}}\}$.

▶ **Example 27** (Continuing example 24). Notice that $\langle \Psi, \sigma \rangle$, where $\Psi = \{(a\ c_1) \curlywedge W, (c_2\ c_1) \curlywedge W\}$ and $\sigma = \{Y \mapsto b, X \mapsto (a\ b) \circ (b\ c) \cdot W\}$, is a solution for $\mathtt{Pr}$.

▶ **Theorem 28** (Correctness). *Let* $\mathtt{Pr}$ *be a unification problem and* $\langle \mathtt{Pr} \rangle_{\mathtt{nf}} = \mathtt{Pr}'$*, then*

1. $\mathcal{U}(\mathtt{Pr}) = \mathcal{U}(\mathtt{Pr}')$*, and*
2. *if* $\mathtt{Pr}'$ *contains equational or inconsistent reduced fixed-point constraints then* $\mathcal{U}(\mathtt{Pr}) = \emptyset$*.*

Proof. The proof is by induction on the length of the derivation $\mathtt{Pr} \overset{n}{\Longrightarrow} \mathtt{Pr}'$.

Base Case. $n = 0$. Then $\mathtt{Pr} = \mathtt{Pr}'$ and the result is trivial.

Induction Step. Suppose, $n > 0$ and consider the reduction chain

$$\mathtt{Pr} = \mathtt{Pr}_1 \Longrightarrow \ldots \Longrightarrow \mathtt{Pr}_{n-1} \Longrightarrow \mathtt{Pr}_n = \mathtt{Pr}'.$$

The proof follows by case analysis on the last rule applied in $\mathtt{Pr}_{n-1}$.

1. The rule is $(\curlywedge at)$. In this case, $\mathtt{Pr}_{n-1} = \mathtt{Pr}'_{n-1} \uplus \{\pi \curlywedge^? a\} \Longrightarrow \mathtt{Pr}'_{n-1} = \mathtt{Pr}_n$, and $\pi(a) = a$.
   Let $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$, then
   a. $\Psi \vdash \pi' \curlywedge t\sigma$, for all $\pi' \curlywedge^? t \in \mathtt{Pr}'_{n-1}$
   b. $\Psi \vdash t\sigma \overset{\curlywedge}{\approx}_\alpha s\sigma$, for all $t \overset{\curlywedge^?}{\approx}_\alpha s \in \mathtt{Pr}'_{n-1}$;
   c. $X\sigma = X\sigma\sigma$, for all $X \in \mathtt{Var}(\mathtt{Pr}'_{n-1})$.
   Therefore, $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_n)$ and $\mathcal{U}(\mathtt{Pr}_{n-1}) \subseteq \mathcal{U}(\mathtt{Pr}_n)$. The other inclusion is trivial.
2. The rule is $(\curlywedge var)$. In this case, $\mathtt{Pr}_{n-1} = \mathtt{Pr}'_{n-1} \uplus \{\pi \curlywedge^? \pi' \cdot X\} \Longrightarrow \mathtt{Pr}'_{n-1} \cup \{\pi^{(\pi')^{-1}} \curlywedge^? X\} = \mathtt{Pr}_n$, and $\pi' \neq Id$.
   Let $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$, then
   a. $\Psi \vdash \pi' \curlywedge t\sigma$, for all $\pi' \curlywedge^? t \in \mathtt{Pr}'_{n-1}$, and $\Psi \vdash \pi \curlywedge \pi' \cdot X\sigma$.
   b. $\Psi \vdash t\sigma \overset{\curlywedge}{\approx}_\alpha s\sigma$, for all $t \overset{\curlywedge^?}{\approx}_\alpha s \in \mathtt{Pr}'_{n-1}$;
   c. $X\sigma = X\sigma\sigma$, for all $X \in \mathtt{Var}(\mathtt{Pr}'_{n-1})$.
   Notice that

$$\begin{aligned} \Psi \vdash \pi \curlywedge \pi' \cdot X\sigma \quad &\Rightarrow \Psi \vdash \pi \cdot (\pi' \cdot X\sigma) \overset{\curlywedge}{\approx}_\alpha (\pi' \cdot X\sigma), \text{ hence} \\ &\Psi \vdash (\pi')^{-1} \circ \pi \circ \pi' \cdot (X\sigma) \overset{\curlywedge}{\approx}_\alpha X\sigma \text{ via Lemma 12} \\ &\Rightarrow \Psi \vdash \pi^{(\pi')^{-1}} \curlywedge X\sigma. \end{aligned}$$

   Therefore, $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_n)$ and $\mathcal{U}(\mathtt{Pr}_{n-1}) \subseteq \mathcal{U}(\mathtt{Pr}_n)$. The other inclusion is similar.
3. The rule is $(\curlywedge \mathbf{abs})$. Then

$$\mathtt{Pr}_{n-1} = \mathtt{Pr}' \uplus \{\pi \curlywedge^? [a]s\} \Longrightarrow \mathtt{Pr}' \cup \{(c_1\ c_2) \curlywedge^? \mathtt{Var}(s), \pi \curlywedge^? (a\ c_1).s\} = \mathtt{Pr}_n.$$

   where c_1 and c_2 are new names not occurring anywhere in the problem.
   Let $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$ be a solution for $\mathtt{Pr}_{n-1}$:
   a. $\Psi \vdash \pi' \curlywedge t\sigma$, for all $\pi' \curlywedge^? t \in \mathtt{Pr}'$ and $\Psi \vdash \pi \curlywedge ([a]s)\sigma$.
   b. $\Psi \vdash t\sigma \overset{\curlywedge}{\approx}_\alpha s\sigma$, for all $t \overset{\curlywedge^?}{\approx}_\alpha s \in \mathtt{Pr}'$.

Since $\Psi \vdash \pi \curlywedge ([a]s)\sigma$ and $([a]s)\sigma = [a]s\sigma$, it follows that $\Psi \vdash \pi \curlywedge ([a]s)\sigma$. From inversion and rule $(\curlywedge[\mathbf{a}])$, this implies that there exists a proof for $\Psi, (c_1\ c_2) \curlywedge \mathtt{Var}(s\sigma) \vdash \pi \curlywedge (a\ c_1).s\sigma$. Notice that we can always choose c_1 and c_2 such that $\mathtt{supp}((c_1\ c_2)) \cap \mathtt{supp}(s\sigma) = \emptyset$, from Lemma 11, it follows that $\Psi \vdash (c_1\ c_2) \curlywedge s\sigma$. Since $\Psi, \overline{(c_1\ c_2) \curlywedge \mathtt{Var}(s\sigma)} \vdash \pi \curlywedge (a\ c_1).s\sigma$, it follows that $\Psi \vdash \pi \curlywedge (a\ c_1).s\sigma$, by Proposition 16. ◀

▶ Remark. Theorem 18 guarantees the equivalence between $\approx_\alpha$ and $\overset{\curlywedge}{\approx}_\alpha$, therefore, we can associate the unification algorithm proposed, with the standard nominal unification algorithm proposed in [16]. The problem $\mathtt{Pr}$ introduced in Example 24, is equivalent to the nominal unification problem $\mathcal{P} = \{[a]\mathsf{f}(X, a) \approx_\alpha{}^? [b]\mathsf{f}((b\ c) \cdot W, (a\ c) \cdot Y\}$, and using the standard simplification rules [16]:

$$\begin{aligned}\mathcal{P} \overset{*}{\Longrightarrow} \overset{[Y\mapsto b]}{\Longrightarrow} \overset{*}{\Longrightarrow} \mathcal{P}' = \{X \approx_\alpha{}^? (a\ b) \cdot ((b\ c) \cdot W), a\#{}^? W\} \\ \overset{[X\mapsto (a\ b)\circ(b\ c)\cdot W]}{\Longrightarrow} \{a\#{}^? W\} = \mathcal{P}'\end{aligned} \tag{5}$$

The pair $\langle \mathcal{P} \rangle_{\mathtt{sol}} = \langle \{a\#W\}, \delta \rangle$, where $\delta = \{Y/b, X \mapsto (a\ b) \circ (b\ c) \cdot W\}$ is a solution for $\mathcal{P}$. Using the translation $[\_]_\curlywedge$, we obtain $[\langle \mathcal{P} \rangle_{\mathtt{sol}}]_\curlywedge = \langle \{[a\#W]_\curlywedge\}, \delta \rangle = \langle (a\ c_a) \curlywedge W, \delta \rangle$, where c_a is a new name, which is equivalent to $\langle (a\ c_a) \curlywedge W, (c_a\ c_1) \curlywedge W, \delta \rangle$, for c_a and c_1 not occurring anywhere in $\mathcal{P}$. Therefore, $[\langle \mathcal{P}_{\mathtt{sol}} \rangle]_\curlywedge$ is a solution for $\mathtt{Pr} = \{[a]\mathsf{f}(X, a) \overset{\curlywedge}{\approx}{}^?_\alpha [b]\mathsf{f}((b\ c) \cdot W, (a\ c) \cdot Y\}$. Similarly, from the solution $\langle \Psi, \sigma \rangle$ proposed in Example 27, we obtain $\langle [\Psi]_\#, \sigma \rangle = \langle a\#W, c_1\#W, c_2\#W, \sigma \rangle$, which is a solution for $\mathcal{P}$.

In the theorem below $\mathtt{Pr}_\curlywedge$ denotes a unification problem w.r.t. $\overset{\curlywedge}{\approx}_\alpha$ and $\curlywedge$, and $\mathcal{P}_\#$ denotes a unification problem w.r.t. $\approx_\alpha$ and $\#$.

▶ **Theorem 29.** *Let $\mathtt{Pr}_\curlywedge$ and $\mathcal{P}_\#$ be unification problems such that $[\mathtt{Pr}_\curlywedge]^\# = \mathcal{P}_\#$ and $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_\curlywedge)$ and $\langle \Delta, \delta \rangle \in \mathcal{U}(\mathcal{P}_\#)$ be solutions for $\mathtt{Pr}_\curlywedge$ and $\mathcal{P}_\#$, respectively. Then*
1. $\langle [\Psi]_\#, \sigma \rangle \in \mathcal{U}(\mathcal{P}_\#)$.
2. $\langle [\Delta]_\curlywedge, \delta \rangle \in \mathcal{U}(\mathtt{Pr}_\curlywedge)$.

5 Nominal C-unification via fixed-point constraints

In this section we propose an approach to nominal unification modulo commutativity via the notion of fixed-point constraints.

For example, assuming $+$ is commutative, i.e., $X + Y = Y + X$, a problem of the form

$$+\langle (a\ b) \cdot X, a \rangle \overset{\curlywedge}{\approx}{}^?_\alpha +\langle Y, X \rangle \tag{6}$$

can be solved by unifying $(a\ b) \cdot X$ with Y and a with X, or $(a\ b) \cdot X$ with X and a with Y.

In [2], a simplification algorithm for solving nominal C-unification was proposed. This algorithm was based on the standard nominal unification algorithm [16] where α-equivalence is defined w.r.t. the notion of freshness. Upon the input of a unification problem $\mathcal{P}$, the algorithm outputs a finite family of triples of the form $\langle \nabla, \sigma, P \rangle$, where ∇ is a freshness context, σ a substitution and P is a set of fixed-point constraints. In [3] we proved that even a simple unification problem, as $(a\ b) \cdot X \approx_\alpha X$ could produce an infinite and independent set of solutions, whenever the signature contains commutative function symbols: $\{X/a + b, X/f(a+b), X/[e]\langle a+b, b+a \rangle, \ldots\}$. Therefore, we could not provide a finite set of solutions consisting only of freshness constraints and substitutions. However, we remark that the problem $+\langle (a\ b) \cdot X, a \rangle \overset{\curlywedge}{\approx}{}^?_\alpha \langle Y, X \rangle$ mentioned above has in fact a finite number of most general solutions (indeed, two) if we solve it using fixed-point constraints. The most general unifiers are $\{X \mapsto a, Y \mapsto b\}$ and $\{Y \mapsto a, (a\ b) \curlywedge X\}$.

$$\frac{\pi(a) = a}{\Upsilon \vdash \pi \curlywedge_C a}\ (\curlywedge_{\mathbf{C}}\mathbf{a}) \qquad \frac{\mathtt{supp}(\pi^{\pi'^{-1}}) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash \pi \curlywedge_C \pi' \cdot X}\ (\curlywedge_{\mathbf{C}}\mathbf{var})$$

$$\frac{\Upsilon \vdash \pi \curlywedge_C t}{\Upsilon \vdash \pi \curlywedge_C \mathsf{f}t}\ \mathsf{f} \neq +\ (\curlywedge_{\mathbf{C}}\mathsf{f}) \qquad \frac{\Upsilon \vdash \pi \cdot t_0 \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t_i \quad \Upsilon \vdash \pi \cdot t_1 \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t_{(i+1) \bmod 2}}{\Upsilon \vdash \pi \curlywedge_C +(t_0, t_1)}\ i = 0, 1 (\curlywedge_{\mathbf{C}}+)$$

$$\frac{\Upsilon \vdash \pi \curlywedge_C t_1 \quad \ldots \quad \Upsilon \vdash \pi \curlywedge_C t_n}{\Upsilon \vdash \pi \curlywedge_C (t_1, \ldots, t_n)}\ (\curlywedge_{\mathbf{C}}\mathbf{tuple}) \qquad \frac{\Upsilon, (c_1\ c_2) \curlywedge_C \mathtt{Var}(t) \vdash \pi \curlywedge_C (a\ c_1) \cdot t}{\Upsilon \vdash \pi \curlywedge_C [a]t}\ (\curlywedge_{\mathbf{C}}\mathbf{abs})$$

Figure 5 Fixed-point rules modulo commutativity.

$$\frac{}{\Upsilon \vdash a \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} a}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathbf{a}) \qquad \frac{\Upsilon \vdash (\pi')^{-1} \circ \pi \curlywedge_C X}{\Upsilon \vdash \pi \cdot X \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} \pi' \cdot X}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathbf{var})$$

$$\frac{\Upsilon \vdash t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t'}{\Upsilon \vdash \mathsf{f}t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} \mathsf{f}t'}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathsf{f}, \mathsf{f} \neq +) \qquad \frac{\Upsilon \vdash t_1 \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t'_1 \quad \ldots \quad \Upsilon \vdash t_n \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t'_n}{\Upsilon \vdash (t_1, \ldots, t_n) \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} (t'_1, \ldots, t'_n)}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathbf{tuple})$$

$$\frac{\Upsilon \vdash t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t'}{\Upsilon \vdash [a]t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} [a]t'}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}[\mathbf{a}]) \qquad \frac{\Upsilon \vdash\ s \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} (a\ b)t \quad \Upsilon, (c_1\ c_2) \curlywedge_C \mathtt{Var}(t) \vdash (a\ c_1) \curlywedge_C t}{\Upsilon \vdash [a]s \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} [b]t}\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathbf{ab})$$

$$\frac{\Upsilon \vdash s_0 \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t_i \quad s_1 \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t_{(i+1) \bmod 2}}{\Upsilon \vdash +\langle s_0, s_1\rangle \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} +\langle t_0, t_1\rangle}\ i = 0, 1\ (\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}} +)$$

Figure 6 Rules for equality modulo commutativity.

▶ **Definition 30** (C-constraints). A C-*fixed-point constraint* is a pair of the form $\pi \curlywedge_C t$, of a permutation π and a term t. A C-α-*equality constraint* (for short, C-equality constraint) is a pair of the form $s \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t$, for nominal terms s and t.

Intuitively, $s \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t$ will mean that s and t are α-equivalent modulo commutativity of some function symbols, and $\pi \curlywedge_C t$ will mean that the permutation π has no effect on term t except for the commutativity of some subterms. For instance, $(a\ c) \curlywedge_C +\langle a, c\rangle$, but not $(a\ c) \curlywedge_C \mathsf{f}\langle a, c\rangle$, if f is not a commutative symbol

The notions of C-fixed-point contexts and C-judgements are defined as expected, and derivable according to the rules in Figures 5 and 6.

Rule $(\curlywedge_{\mathbf{C}}\mathbf{var})$ is similar to the previous one. Rule $(\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}}\mathbf{var})$ relies on the primitive notion of fixed-point constraints, it is equivalent to the rule given earlier. There is a branching rule $(\curlywedge_{\mathbf{C}}+)$ for C-fixed-point constraints and a branching rule $(\mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathbf{C}} +)$ for C-equality constraints (more precisely, in the case of C operators, there are two possible rules to apply, but we have written them in a compact way as one rule with parameter i). Technical results proven in Section 3 can be extended to C-constraints.

▶ **Theorem 31.** *Let Υ, π and t be a C-fixed-point context, a permutation and a nominal term, respectively. $\Upsilon \vdash \pi \curlywedge_C t$ iff $\Upsilon \vdash \pi \cdot t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,C} t$.*

Proof. The proof is by induction on the structure of t, and follows the same lines of the proof of Theorem 18. ◀

$$\frac{\nabla \vdash s \approx_{\{\alpha,C\}} t}{\nabla \vdash f_k^E\ s \approx_{\{\alpha,C\}} f_k^E\ t}\ ,\quad E \neq C \text{ or both } s \text{ and } t \text{ are not pairs}\quad (\approx_{\{\alpha,\mathbf{C}\}} \mathbf{app})$$

$$\frac{\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_i,\quad \nabla \vdash s_1 \approx_{\{\alpha,C\}} t_{(i+1)\,mod\,2}}{\nabla \vdash f_k^C\ \langle s_0, s_1\rangle \approx_{\{\alpha,C\}} f_k^C\ \langle t_0, t_1\rangle}\ ,\quad i = 0,1\quad (\approx_{\{\alpha,\mathbf{C}\}} \mathbf{C})$$

Figure 7 Additional rules for $\{\alpha, C\}$-equivalence.

5.1 From freshness to C-fixed-point constraints

In [2] the relation $\approx_{\{\alpha,C\}}$ was defined as an extension of $\approx_\alpha$ (see the rules in Figures 3 and 4) with rules for commutative symbols:

Using the functions $[\_]_{\curlywedge}$ and $[\_]_{\#}$ defined in Section 3.2, we can obtain results that extend Lemma 19 and Theorem 20.

▶ **Lemma 32.** $\Delta \vdash a\#t \Rightarrow [\Delta]^C_{\curlywedge}, (c_1\ c_2) \curlywedge_C \mathtt{Var}(t) \vdash (a\ c_1) \curlywedge_C t$,
where $[\Delta]^C_{\curlywedge} = \{\pi \curlywedge_C X \mid \pi \curlywedge X \in [\Delta]_{\curlywedge}\}$.

▶ **Theorem 33.**

1. $\Upsilon \vdash s \overset{\curlywedge}{\approx}_{\alpha,C} t \Rightarrow [\Upsilon]_{\#} \vdash s \approx_{\{\alpha,C\}} t$.
2. $\Delta \vdash s \approx_{\{\alpha,C\}} t \Rightarrow [\Delta]^C_{\curlywedge} \vdash s \overset{\curlywedge}{\approx}_{\alpha,C} t$.

5.2 Solving nominal C-unification problems via fixed-point constraints

Similarly to Section 4, we define the notion of nominal C-unification in terms of C-fixed-point constraints.

▶ **Definition 34.** A C-*unification problem* $\mathtt{Pr}$ consists of a finite set of C-equality and C-fixed-point constraints of the form $s \overset{\curlywedge?}{\approx}_C t$ and $\pi \curlywedge^?_C t$, respectively[3].

We write $\mathtt{Pr} \Longrightarrow_C \mathtt{Pr}'$ when $\mathtt{Pr}'$ is obtained from $\mathtt{Pr}$ by applying a simplification rule from Table 2 and we write $\overset{*}{\Longrightarrow}_C$ for the reflexive and transitive closure of $\Longrightarrow_C$. We omit the subindex when it is clear from the context.

▶ **Lemma 35.** *There is no infinite chain of reductions* $\Longrightarrow_C$ *starting from a C-unification problem* $\mathtt{Pr}$.

The simplification rules (Table 2) specify a C-*unification algorithm*: we apply the simplification rules in a problem $\mathtt{Pr}$ until we reach a normal form $\langle \mathtt{Pr}\rangle_{\mathtt{nf}}$. The notions of *solution, consistency, failure, correctness*, etc. obtained in Section 4 can be extended to C-unification.

▶ Remark. As with standard nominal unification, one can use the functions $[\_]_{\#}$ and $[\_]_{\curlywedge}$ to represent solutions $\langle \nabla, \sigma, P\rangle$ of nominal C-unification problems w.r.t. freshness constraints [2, 3] (where P is a set of fixed-point equations of the form $\pi.X \approx^?_{\{\alpha,C\}} X$) as solutions $\langle [\nabla]_{\curlywedge} \cup \{P_{\curlywedge_C}\}, \sigma\rangle$ of nominal C-unification problems via C-fixed-point constraints, where $P_{\curlywedge_C} = \{\pi \curlywedge_C X \mid \pi.X \approx^?_{\{\alpha,C\}} X \in P\}$.

[3] To ease the notation, we will denote $s \overset{\curlywedge?}{\approx}_C t$ by $s \approx^? t$.

Table 2 Simplification Rules for C-unification problems. In rules $(\curlywedge_C abs)$ and $(\overset{\curlywedge}{\approx}_{\alpha,C} abs2)$, c_1 and c_2 are new names.

$(\curlywedge_C\ at)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? a\}$	$\Longrightarrow$	$\mathtt{Pr}$, if $\pi(a) = a$
$(\curlywedge_C f)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? \mathtt{f}t\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge_C^? t\}, \mathtt{f} \neq +$
$(\curlywedge_C + 1)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? +\langle t_0, t_1\rangle\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \cdot t_0 \approx^? t_0, \pi \cdot t_1 \approx^? t_1\}$
$(\curlywedge_C + 2)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? +\langle t_0, t_1\rangle\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \cdot t_0 \approx^? t_1, \pi \cdot t_1 \approx^? t_0\}$
$(\curlywedge_C\ tuple)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? (\widetilde{t})_n\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge_C^? t_1, \ldots, \pi \curlywedge_C^? t_n\}$
$(\curlywedge_C abs)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? [a]t\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi \curlywedge_C^? (a\ c_1) \cdot t, \overline{(c_1\ c_2) \curlywedge_C^? \mathtt{Var}(t)}\}$
$(\curlywedge_C var)$	$\mathtt{Pr} \uplus \{\pi \curlywedge_C^? \pi' \cdot X\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{\pi^{(\pi')^{-1}} \curlywedge_C^? X\}$, if $\pi' \neq Id$
$(\overset{\curlywedge}{\approx}_{\alpha,C} a)$	$\mathtt{Pr} \uplus \{a \approx^? a\}$	$\Longrightarrow$	$\mathtt{Pr}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} f)$	$\mathtt{Pr} \uplus \{\mathtt{f}t \approx^? \mathtt{f}t'\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \approx^? t'\}, \mathtt{f} \neq +$
$(\overset{\curlywedge}{\approx}_{\alpha,C} +1)$	$\mathtt{Pr} \uplus \{+\langle t_0, t_1\rangle \approx^? +\langle s_0, s_1\rangle\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t_0 \approx^? s_0, t_1 \approx^? s_1\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} +2)$	$\mathtt{Pr} \uplus \{+\langle t_0, t_1\rangle \approx^? +\langle s_0, s_1\rangle\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t_0 \approx^? s_1, t_1 \approx^? s_0\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} t)$	$\mathtt{Pr} \uplus \{(\widetilde{t})_n \approx^? (\widetilde{t'})_n\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t_1 \approx^? t'_1, \ldots, t_n \approx^? t'_n\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} abs1)$	$\mathtt{Pr} \uplus \{[a]t \approx^? [a]t'\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \approx^? t'\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} abs2)$	$\mathtt{Pr} \uplus \{[a]t \approx^? [b]s\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{t \approx^? (a\ b) \cdot s, \overline{(a\ c_1) \curlywedge_C^? s, (c_1\ c_2) \curlywedge_C^? \mathtt{Var}(s)}\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} var)$	$\mathtt{Pr} \uplus \{\pi \cdot X \approx^? \pi' \cdot X\}$	$\Longrightarrow$	$\mathtt{Pr} \cup \{(\pi')^{-1} \circ \pi \curlywedge_C^? X\}$
$(\overset{\curlywedge}{\approx}_{\alpha,C} inst1)$	$\mathtt{Pr} \uplus \{\pi \cdot X \approx^? t\}$	$\overset{[X \mapsto \pi^{-1}.t]}{\Longrightarrow}$	$\mathtt{Pr}\{X \mapsto \pi^{-1}.t\}$, if $X \notin \mathtt{Var}(t)$
$(\approx_\alpha inst2)$	$\mathtt{Pr} \uplus \{t \approx^? \pi \cdot X\}$	$\overset{[X \mapsto \pi^{-1}.t]}{\Longrightarrow}$	$\mathtt{Pr}\{X \mapsto \pi^{-1}.t\}$, if $X \notin \mathtt{Var}(t)$

6 Conclusions and Future Work

The notion of fixed-point constraints allowed us to obtain a finite representation of solutions for nominal C-unification problems. This brings a novel alternative to standard nominal unification approaches in which just the algebra of atom permutations and the logic of freshness constraints are used to implement equational reasoning (e.g., [1, 5, 6, 7, 9]), and in particular to their extensions modulo commutativity, for which only infinite representations were possible in the standard approach. With the new proposed approach the development of algorithms for the generation of solutions of nominal equational problems modulo theories such as C, AC, etc would be simplified avoiding with the use of fixed-point constraints the development of procedures for the generation of infinite independent sets of solutions.

In future work we plan to extend this approach to matching and unification modulo different equational theories as well as to the treatment of equational problems in nominal rewriting modulo.

References

1 T. Aoto and K. Kikuchi. *A Rule-Based Procedure for Equivariant Nominal Unification*. In *Pre-proc. of Higher-Order Rewriting (HOR)*, pages 1–5, 2016.

2 M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. *Nominal C-Unification*. In *Pre-proc. of the 27th Int. Symp. Logic-based Program Synthesis and Transformation (LOPSTR)*, pages 1–15, 2017. URL: `https://arxiv.org/abs/1709.05384`.

3 M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. *On Solving Nominal Fixpoint Equations*. In *Proc. of the 11th Int. Symp. on Frontiers of Combining*

Systems (FroCoS), volume 10483 of *Lecture Notes in Computer Science*, pages 209–226. Springer, 2017. doi:10.1007/978-3-319-66167-4_12.

4 M. Ayala-Rincón, W. de Carvalho Segundo, M. Fernández, and D. Nantes-Sobrinho. A formalisation of nominal α-equivalence with A and AC function symbols. *Electronic Notes in Theoretical Computer Science*, 332:21–38, 2017. doi:10.1016/j.entcs.2017.04.003.

5 C. F. Calvès. Unifying Nominal Unification. In *24th International Conference on Rewriting Techniques and Applications (RTA 2013)*, volume 21 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 143–157, 2013. doi:10.4230/LIPIcs.RTA.2013.143.

6 C. F. Calvès and M. Fernández. A Polynomial Nominal Unification Algorithm. *Theoretical Computer Science*, 403(2-3):285–306, 2008. doi:10.1016/j.tcs.2008.05.012.

7 J. Cheney. *Equivariant unification. Journal of Automated Reasoning*, 45(3):267–300, 2010. doi:10.1007/s10817-009-9164-3.

8 M. Fernández and M. J. Gabbay. *Nominal Rewriting. Information and Computation*, 205(6):917–965, 2007. doi:10.1016/j.ic.2006.12.002.

9 M. Fernández, M. J. Gabbay, and I. Mackie. *Nominal Rewriting Systems.* In *Proc. of the 6th Int. Conf. on Principles and Practice of Declarative Programming (PPDP)*, pages 108–119. ACM Press, 2004. doi:10.1145/1013963.1013978.

10 M. J. Gabbay. *A Theory of Inductive Definitions With α-equivalence.* PhD thesis, DPMMS and Trinity College, University of Cambridge, 2000.

11 M. J. Gabbay and A. Mathijssen. Capture-avoiding substitution as a nominal algebra. *Formal Aspects of Computing*, 20(4-5):451–479, 2008. doi:10.1007/s00165-007-0056-1.

12 M. J. Gabbay and A. M. Pitts. *A New Approach to Abstract Syntax with Variable Binding. Formal Aspects of Computing*, 13(3-5):341–363, 2002. doi:10.1007/s001650200016.

13 A. M. Pitts. Nominal Logic, a First Order Theory of Names and Binding. *Information and Computation*, 186(2):165–193, 2003. doi:10.1016/S0890-5401(03)00138-X.

14 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge University Press, 2013.

15 M. Schmidt-Schauß, T. Kutsia, J. Levy, and M. Villaret. *Nominal Unification of Higher Order Expressions with Recursive Let.* In *26th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR), Revised Selected Papers*, volume 10184 of *Lecture Notes in Computer Science*, pages 328–344, 2016. doi:10.1007/978-3-319-63139-4_19.

16 C. Urban, A. M. Pitts, and M. J. Gabbay. *Nominal Unification. Theoretical Computer Science*, 323(1-3):473–497, 2004. doi:10.1016/j.tcs.2004.06.016.