Logical Methods in Computer Science
Volume 16, Issue 1, 2020, pp. 19:1–19:36
https://lmcs.episciences.org/

Submitted Feb. 25, 2019
Published Feb. 17, 2020

# ON NOMINAL SYNTAX AND PERMUTATION FIXED POINTS

MAURICIO AYALA-RINCÓN, MARIBEL FERNÁNDEZ, AND DANIELE NANTES-SOBRINHO

Departments Mathematics and Computer Science, Universidade de Brasília, Brasília, Brazil
*e-mail address*: ayala@unb.br

Department of Informatics, King's College London, London, UK
*e-mail address*: maribel.fernandez@kcl.ac.uk

Department Mathematics, Universidade de Brasília, Brasília, Brazil
*e-mail address*: dnantes@unb.br

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 this is not the case when unifiers are expressed using freshness contexts. We provide a definition of $\alpha$-equivalence modulo equational theories that takes into account $\mathsf{A}$, $\mathsf{C}$ and $\mathsf{AC}$ theories. Based on this notion of equivalence, we show that $\mathsf{C}$-unification is finitary and we provide a sound and complete $\mathsf{C}$-unification algorithm, as a first step towards the development of nominal unification modulo $\mathsf{AC}$ and other equational theories with permutative properties.

## 1. Introduction

This paper presents a new approach for the definition of nominal languages, based on the use of permutation fixed points. More precisely, we give a new axiomatisation of the $\alpha$-equivalence relation for nominal terms using permutation fixed-points, and revisit nominal unification in this setting.

In nominal syntax [UPG04], *atoms* are used to represent object-level variables and *atom permutations* are used to implement renamings, following the nominal-sets approach advocated by Gabbay and Pitts [Gab00, GP02, Pit13]. Atoms can be abstracted over terms; the syntax $[a]s$ represents the abstraction of $a$ in $s$. To rename an abstracted atom $a$ to $b$,

*Key words and phrases:* alpha-conversion, nominal syntax, unification, equational theories.

Work supported by FAPDF grant 193001369/2016. M. Ayala-Rincón partially funded by CNPq research grant number 307672/2017-4.


DOI:10.23638/LMCS-16(1:19)2020

a *swapping* permutation $\pi = (a\,b)$ is applied. Thus, the action of $\pi$ over $[a]s$, written as $(a\,b) \cdot [a]s$, produces the nominal term $[b]s'$, where $s'$ is the result of swapping $a$ and $b$ in $s$. The $\alpha$-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 $\alpha$-equivalence requires another first-order specialised calculus to check freshness constraints. For instance, checking whether $[a]s \approx_\alpha [b]t$ reduces to checking 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 $\pi$ and $\rho$ (i.e., the set $\{a : \pi(a) \neq \rho(a)\}$) are fresh in $s$.

If the domain of a permutation $\pi$ 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* [UPG04], 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 $\mathsf{C}$-nominal unification (nominal unification modulo commutativity), to specify that a permutation has no effect on the instances of $X$ modulo $\mathsf{C}$, in other words, to specify that the permutation does not affect a given $\mathsf{C}$-equivalence class, we need something more than a freshness constraint (note that $(a\ b) \cdot (a + b) = b + a =_\mathsf{C} a + b$, 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 $\alpha$-equivalence of nominal terms using permutation fixed-point constraints: we write $\pi \curlywedge t$ (read "$\pi$ fixes $t$") if $t$ is a fixed-point of $\pi$. We show how to derive fixed-point constraints $\pi \curlywedge t$ from primitive constraints of the form $\pi \curlywedge X$, and show the correctness of this approach by proving that the $\alpha$-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 specify $\alpha$-equivalence modulo equational theories containing $\mathsf{A}$, $\mathsf{C}$, and $\mathsf{AC}$ operators, and provide an algorithm to solve nominal unification problems modulo $\mathsf{C}$, which outputs a finite set of most general solutions.

**Related Work.** Equational reasoning has been extensively explored since the early development of modern abstract algebra (see, e.g., the $E$-unification surveys by Siekmann [Sie90] and Baader et al [BS94] and [BSN$^+$01]). For $\mathsf{AC}$-equality checking, $\mathsf{AC}$ matching and $\mathsf{AC}$ unification, refined techniques have been applied. For instance, $\mathsf{AC}$-equality check and linear $\mathsf{AC}$-matching problems can be reduced to searching a perfect matching in a bipartite graph [BKN87], whereas $\mathsf{AC}$ unification problems can be reduced to solving a system of Diophantine equations [Sti81, Fag87].

Techniques to deal with $\alpha$-equivalence modulo the equational theories $\mathsf{A}$, $\mathsf{C}$ and $\mathsf{AC}$ were proposed in [AdCSFN17, AdCSMFR19, ARdCSFNS17, ARdCSFNS18], using the standard nominal approach via freshness constraints. Solving nominal $\mathsf{C}$-unification problems

requires to deal with fixed-point equations, for which there is no finitary representation of the set of solutions using only freshness constraints and substitutions [ARdCSFNS17, ARdCSFNS18]. A combinatorial algorithm permits to find all the solutions of fixed-point equations [ARdCSFNS18].

Fixed-point constraints arise also in other contexts: Schmidt-Schauß et al [SKLV17] show how nominal unification problems in a language with recursive let operators give rise to freshness constraints and nominal fixed-point equations. The approach to nominal unification via permutation fixed-points proposed in this paper could also be used to reason about equality in this language.

This paper is a revised and extended version of [AFN18], where nominal terms with fixed-point permutation constraints were first presented. In this paper we prove the correctness of the approach, by showing it is equivalent to the standard presentation via freshness constraints. We also provide proofs of soundness and completeness of the unification algorithm. A generalisation of the notion of $\alpha$-equivalence to take into account equational theories including $\mathsf{A}$, $\mathsf{C}$ and $\mathsf{AC}$ is also provided (only $\mathsf{C}$ was considered in [AFN18]), as well as a $\mathsf{C}$-unification algorithm.

**Overview.** The paper is organised as follows. Section 2 provides the necessary background material on nominal syntax and semantics. Section 3 introduces fixed-point constraints and $\alpha$-equivalence, and shows that these relations behave as expected. In particular, we show that there is a two-way translation between the freshness-based $\alpha$-equivalence relation and its permutation fixed-point counter-part, which confirms that the fixed-point approach is equivalent to the standard approach via freshness constraints. Section 4 presents a nominal unification algorithm specified by a set of simplification rules, and proves its soundness and completeness. In Section 5 we generalise the approach to take into account equational theories: we define a notion of permutation fixed-point $\alpha$-equivalence modulo $\mathsf{A}$, $\mathsf{C}$ and $\mathsf{AC}$ theories, and develop a $\mathsf{C}$-unification algorithm. Finally, Section 6 concludes and discusses future work.

## 2. Preliminaries

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 2.1** (Nominal grammar)**.** Nominal terms are generated by the following grammar.

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

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, function symbols are equipped with an equational theory $\mathsf{E}$, hence $\mathsf{f}^{\mathsf{E}}\ t$ denotes the *application of* $\mathsf{f}^{\mathsf{E}}$ *to* $t$ and $\pi \cdot X$ is a *moderated variable* or *suspension*, where $\pi$ is an atom permutation. We write $\mathsf{f}^{\emptyset}$, or simply $\mathsf{f}$, to emphasise that no equational theory is assumed for $\mathsf{f}$, that is, $\mathsf{f}$ is just a function symbol.

We follow the *permutative convention* [GM08, 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 $\pi$ is generated by the following grammar:

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

where $Id$ is the identity permutation, usually omitted from the list of swappings representing a permutation $\pi$. Suspensions of the form $Id \cdot X$ will be represented just by $X$. We write $\pi^{-1}$ for the *inverse* of $\pi$, 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 $\pi$, $\pi'$ 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 2.2** (Permutation action)**.** The action of a permutation $\pi$ on a term $t$ is defined by induction on the number of swappings in $\pi$:

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

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

**Definition 2.3** (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

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

The following well-known property of substitution and permutation justifies the notation $\pi \cdot s\sigma$ (without brackets), see [UPG04] for more details.

**Lemma 2.4.** *Substitution and permutation commute:* $\pi \cdot (s\sigma) = (\pi \cdot s)\sigma$.

2.1. **Alpha-equivalence via freshness constraints.** In the standard nominal approach (see, e.g., [Pit03, UPG04, FG07]), the $\alpha$-equivalence relation $s \approx_\alpha t$ is defined 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. These relations are axiomatised using the rules in Figures 1 and 2, respectively.

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

$$\frac{\Delta \vdash a\#t_1 \quad \ldots \quad \Delta \vdash a\#t_n}{\Delta \vdash a\#(t_1, \ldots, t_n)}\ (\#\mathbf{tuple}) \qquad \frac{}{\Delta \vdash a\#[a]t}\ (\#[\mathbf{a}]) \qquad \frac{\Delta \vdash a\#t}{\Delta \vdash a\#[b]t}\ (\#\mathbf{abs})$$

Figure 1: Rules for freshness

We call $s \approx_\alpha t$ and $a\#t$ $\alpha$-equality and freshness constraints, respectively. Note that to define $\approx_\alpha$ we use the difference set of two permutations in rule ($\approx_\alpha$ **var**); we denote by $\mathtt{ds}(\pi, \pi')\#X$ the following set of freshness constraints:

$$\mathtt{ds}(\pi, \pi')\#X = \{a\#X \mid a \in \mathtt{ds}(\pi, \pi')\}.$$

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

$$\frac{\Delta \vdash t \approx_\alpha t'}{\Delta \vdash \mathsf{f}\ t \approx_\alpha \mathsf{f}\ t'}\ (\approx_\alpha \mathsf{f}) \qquad \frac{\Delta \vdash t_1 \approx_\alpha t'_1 \quad \ldots \quad \Delta \vdash t_n \approx_\alpha t'_n}{\Delta \vdash (t_1, \ldots, t_n) \approx_\alpha (t'_1, \ldots, t'_n)}\ (\approx_\alpha \mathbf{tuple})$$

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

Figure 2: Rules for $\alpha$-equality via freshness

The symbols $\Delta$ and $\nabla$ denote *freshness contexts*, which are sets of freshness constraints of the form $a\#X$. The domain of a freshness context $\Delta$, denoted by $\mathtt{dom}(\Delta)$, consists of the atoms occurring in $\Delta$; $\Delta|_X$ consists of the restriction of $\Delta$ to the freshness constraints on variable $X$, that is, the set $\{a\#X \mid a\#X \in \Delta\}$.

2.2. **Nominal sets and support.** Let $S$ be a set equipped with an action of the group $\mathtt{Perm}(\mathbb{A})$ of finite permutations of $\mathbb{A}$.

**Definition 2.5.** A set $A \subset \mathbb{A}$ is a *support* for an element $x \in S$ if for all $\pi \in \mathtt{Perm}(\mathbb{A})$, the following holds

$$((\forall a \in A)\ \pi(a) = a) \Rightarrow \pi \cdot x = x \tag{2.1}$$

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

As in [Pit13], we denote by $\mathtt{supp}_S(x)$ the least finite support of $x$, that is,

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

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

2.3. **The "new" quantifier.** In Nominal Logic [Pit03], the "new" quantifier $\mathsf{И}$ is used to quantify over new names. Given two elements $x$, $y$ of a nominal set, write $x\#y$ as an abbreviation of $\mathtt{supp}(x) \cap \mathtt{supp}(y) = \emptyset$; then $\mathsf{И}a.P(a, x)$ (read "for some/any new atom $a$, $P(a, x)$") abbreviates $\forall a \in \mathbb{A}.a\#x \Rightarrow P(a, x)$ or equivalently $\exists a \in \mathbb{A}.a\#x \wedge P(a, x)$ (see Theorem 3.9 in [Pit13] for more details).

Intuitively, the "new" quantifier is used to indicate that a predicate holds for some (any) new atoms. The formula $\mathsf{И}a.\phi$ means "for a fresh name $a$, $\phi$ holds". Since there is an infinite supply of names and elements of nominal sets have a finite support, it is always possible to pick a fresh name. Moreover, it can be shown that if a property $\phi(a)$ holds for some fresh name $a$ then it holds for all fresh names, so the way in which the fresh atom is chosen is not important.

In proof systems for Nominal Logic, several approaches have been proposed to define rules to introduce and eliminate the И quantifier (see, e.g., [Pit03, Che05, CU08]). We follow Cheney's approach [Che05], where the formula $И a.\phi$ (read "$a$ is a *newly quantified atom* in $\phi$") is well formed if $\phi$ is a well-formed formula and $a \in \mathbb{A}$ is semantically fresh. In other words, the introduction of a И quantifier for $a$ in a formula $\phi$ requires $a$ to be a new atom, fresh for all the variables in $\phi$ (see [Che05], Figure 5, where the well-formedness rules are given). We implicitly assume that $a$ is a newly generated atom when we introduce И in our formulas in the next section. Operationally, proving a formula $И a.\phi$ boils down to picking a fresh atom $c$ and proving that $\phi$ holds when the occurrences of $a$ are replaced by the fresh atom $c$.

## 3. Fixed-point Constraints

The native notion of equality on nominal terms is $\alpha$-equivalence, written $s \approx_\alpha t$. As mentioned in section 2.1, this relation is usually axiomatised using the *freshness relation* between atoms and terms. The notion of freshness is derived from the notion of support, which in turn is defined using permutation fixed-points (see Definition 2.5). We can define freshness using the quantifier И combined with a notion of fixed-point, as shown by Pitts [Pit13] (page 53):

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

In this paper, instead of defining $\alpha$-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 permutation fixed points 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 $\pi \in \mathtt{Perm}(\mathbb{A})$:

**Definition 3.1** (Fixed-point relation)**.** Let $S$ be a nominal set. The *fixed-point relation* $\curlywedge \subseteq \mathtt{Perm}(\mathbb{A}) \times S$ is defined as: $\pi \curlywedge x \Leftrightarrow \pi \cdot x = x$. Read "$\pi \curlywedge x$" as *"$\pi$ fixes $x$"*.

It is easy to see that if $\mathtt{dom}(\pi) \cap \mathtt{supp}(x) = \emptyset$ then $\pi \curlywedge x$ holds (this follows directly from the definition of support, see Definition 2.5). However, the other direction does not hold in general. For example, consider expressions built using atoms and a binary commutative operator $+$; the permutation $\pi = (a\ b)$ fixes the equivalence class of the expression $a + b$ despite the fact that its support coincides with $\mathtt{dom}(\pi)$.

Permutation fixed-points will play an important role in the definition of $\alpha$-equality of nominal terms. Below we define *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, for $s$ and $t$ ground nominal terms

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

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

In the case of non-ground terms, a fixed-point or $\alpha$-equality constraint has to be evaluated in a context, which provides information about permutations that fix the variables.

**Definition 3.2** (Fixed-point and equality constraints)**.** A *fixed-point constraint* is a pair $\pi \curlywedge t$ of a permutation $\pi$ and a term $t$. An *$\alpha$-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 finite 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 $\{\pi \curlywedge X \mid X \in \mathtt{Var}(t)\}$.

We now introduce some notation:

The set $\mathtt{Var}(\Upsilon)$ of variables is defined as expected: it contains all the variables mentioned in the fixed-point context $\Upsilon$.

The set of permutations of a fixed-point context $\Upsilon$ 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 $\sigma$ and a fixed-point context $\Upsilon$ we define $\Upsilon\sigma := \{\pi \curlywedge X\sigma \,|\, \pi \curlywedge X \in \Upsilon\}$.

To axiomatise the relation $\curlywedge$, we rely on the notion of *conjugation* of permutations. Conjugacy plays an important role in group theory [Hun80]. Recall that in a group $G$, the element $b$ is a conjugate of $a$ if there exists an element $g$ such that $gag^{-1} = b$. It can be easily shown that conjugacy is an equivalence relation: if $b$ is a conjugate of $a$ then $a$ is a conjugate of $b$. The conjugacy class of $a$ consists of the elements of the form $gag^{-1}$, for $g \in G$. In the case of permutation groups, the conjugate of $\pi$ with respect to $\rho$, denoted as $\pi^\rho$, is the result of the composition: $\rho \circ \pi \circ \rho^{-1}$.

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

The notion of *support* of a permutation or nominal term will be necessary for the next results. Considering permutations as elements of a nominal set, it follows from Definition 2.5 that $\mathtt{supp}(\pi) = \mathtt{dom}(\pi)$; similarly, in the case of a ground term $t$, the support coincides with its set of free names, i.e., $\mathtt{supp}(t) = \mathtt{fn}(t)$. In the case a term $t$ is not ground, it should be considered in a context, say $\Upsilon$, which gives information about its variables. We define it after introducing fixed-point and $\alpha$-equality judgements.

**Definition 3.3** (Judgements)**.** A *fixed-point judgement* is a tuple $\Upsilon \vdash \mathit{И}\overline{c}.\ \pi \curlywedge t$ of a fixed-point context and a fixed-point constraint where the atoms $\overline{c}$ are newly quantified. Here $\overline{c} = c_1, \ldots, c_n$, and $n = 0$ stands for empty quantification; in that case we may omit the $\mathit{И}$ quantifier and write simply $\Upsilon \vdash \pi \curlywedge t$.

An *$\alpha$-equivalence judgement* is a tuple $\Psi \vdash \mathit{И}\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t$ of a fixed-point context and an equality constraint where the atoms $\overline{c}$ are newly quantified.

The derivable fixed-point and $\alpha$-equivalence judgements are defined by the rules in Figures 3 and 4. We give an example before explaining the rules in more detail.

**Example 3.4.** The term $[a]\mathtt{f}a$ is a fixed-point for the permutation $(a\ b)$, since $(a\ b) \cdot [a]\mathtt{f}a = [b]\mathtt{f}b$, which is $\alpha$-equivalent to $[a]\mathtt{f}a$. As expected, we can derive $\vdash (a\ b) \curlywedge [a]\mathtt{f}a$. For this we

use the rule ($\curlywedge\mathbf{abs}$), which requires to prove $\vdash \mathit{И} c_1.(a\ b) \curlywedge (a\ c_1) \cdot \mathsf{f}a$. Since $(a\ c_1) \cdot \mathsf{f}a = \mathsf{f}c_1$, we need to prove $\vdash \mathit{И} c_1.(a\ b) \curlywedge c_1$, which holds by ($\curlywedge\mathbf{a}$). However, $\mathsf{f}a$ is not a fixed-point for $(a\ b)$: we cannot derive $\vdash (a\ b) \curlywedge \mathsf{f}a$. This is to be expected, since $\mathsf{f}a$ and $\mathsf{f}b$ are not $\alpha$-equivalent; we cannot derive $\vdash (a\ b) \cdot \mathsf{f}a \mathrel{\overset{\curlywedge}{\approx}_\alpha} \mathsf{f}a$.

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

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

$$\frac{\Upsilon \vdash \mathit{И}\overline{c}, c_1.\pi \curlywedge (a\ c_1) \cdot t}{\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge [a]t}\ (\curlywedge\mathbf{abs})$$

Figure 3: Fixed-point rules.

The premise of rules ($\curlywedge\mathbf{abs}$) and ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{ab}$) introduces a newly quantified atom $c_1$. Recall that the introduction of a $\mathit{И}$ quantifier requires the use of a new atom by well formedness (see Section 2.3). Therefore, in rules ($\curlywedge\mathbf{abs}$) and ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{ab}$) we can assume that the atom $c_1$ does not occur in the conclusion.

In rule ($\curlywedge\mathbf{var}$), the condition $\mathtt{supp}(\pi^{\pi'^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ means that the permutation $\pi^{\pi'^{-1}}$ can only affect atoms in $\mathtt{perm}(\Upsilon|_X)$ and new atoms, 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 $\pi$ if $\pi \cdot [a]t$ is $\alpha$-equivalent to $[a]t$, that is, $[\pi(a)]\pi \cdot t$ is $\alpha$-equivalent to $[a]t$; the latter means that the only free atom in $t$ that could be affected by $\pi$ is $a$, hence, if we replace occurrences of $a$ in $t$ with another, new atom $c$, $\pi$ should have no effect. Note that if a judgement $\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge t$ is derivable, then so is $\Upsilon \vdash \mathit{И}\overline{c}, \overline{c'}.\pi \curlywedge t$ for a new set of atoms $\overline{c'}$ (there is an infinite supply of new atoms).

$$\frac{}{\Upsilon \vdash \mathit{И}\overline{c}.\ a \mathrel{\overset{\curlywedge}{\approx}_\alpha} a}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{a}) \qquad \frac{\mathtt{supp}((\pi')^{-1} \circ \pi) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash \mathit{И}\overline{c}.\ \pi \cdot X \mathrel{\overset{\curlywedge}{\approx}_\alpha} \pi' \cdot X}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{var})$$

$$\frac{\Upsilon \vdash \mathit{И}\overline{c}.\ t \mathrel{\overset{\curlywedge}{\approx}_\alpha} t'}{\Upsilon \vdash \mathit{И}\overline{c}.\ \mathsf{f}\ t \mathrel{\overset{\curlywedge}{\approx}_\alpha} \mathsf{f}\ t'}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{f}) \qquad \frac{\Upsilon \vdash \mathit{И}\overline{c}.t_1 \mathrel{\overset{\curlywedge}{\approx}_\alpha} t'_1 \quad \ldots \quad \Upsilon \vdash \mathit{И}\overline{c}.t_n \mathrel{\overset{\curlywedge}{\approx}_\alpha} t'_n}{\Upsilon \vdash \mathit{И}\overline{c}.(t_1, \ldots, t_n) \mathrel{\overset{\curlywedge}{\approx}_\alpha} (t'_1, \ldots, t'_n)}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{tuple})$$

$$\frac{\Upsilon \vdash \mathit{И}\overline{c}.\ t \mathrel{\overset{\curlywedge}{\approx}_\alpha} t'}{\Upsilon \vdash \mathit{И}\overline{c}.\ [a]t \mathrel{\overset{\curlywedge}{\approx}_\alpha} [a]t'}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}[\mathbf{a}]) \qquad \frac{\Upsilon \vdash \mathit{И}\overline{c}.\ s \mathrel{\overset{\curlywedge}{\approx}_\alpha} (a\ b) \cdot t \qquad \Upsilon \vdash \mathit{И}\overline{c}, c_1.(a\ c_1) \curlywedge t}{\Upsilon \vdash \mathit{И}\overline{c}.\ [a]s \mathrel{\overset{\curlywedge}{\approx}_\alpha} [b]t}\ (\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathbf{ab})$$

Figure 4: Rules for $\alpha$-equality.

The $\alpha$-equivalence relation is defined using fixed-point judgements (see rule ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{ab}$)). Rules ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{a}$), ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{f}$), ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}[\mathsf{a}]$) and ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{tuple}$) are defined as expected, whereas the intuition behind rule ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{var}$) is similar to the corresponding rule in Figure 3. The most interesting rule is ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}\mathsf{ab}$). 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 newly generated atom $c_1$.

**Example 3.5.** Notice that $(b\,e) \curlywedge X \vdash (a\,b) \curlywedge [a](\mathsf{f}a, (c\,d) \cdot X)$. This is derived as follows: by rule ($\curlywedge\mathbf{abs}$) we have to show $(b\,e) \curlywedge X \vdash \mathit{И}g.(a\,b) \curlywedge (\mathsf{f}g, (a\,g)(c\,d) \cdot X)$; by rule ($\curlywedge\mathbf{tuple}$) this requires $(b\,e) \curlywedge X \vdash \mathit{И}g.(a\,b) \curlywedge \mathsf{f}g$ and $(b\,e) \curlywedge X \vdash \mathit{И}g.(a\,b) \curlywedge (a\,g)(c\,d) \cdot X$; the former holds by application of rules ($\curlywedge\mathsf{f}$) and ($\curlywedge\mathsf{a}$) since $(a\,b) \cdot g = g$, and the latter by rule ($\curlywedge\mathbf{var}$), since $\mathtt{supp}((a\ b)^{(c\ d)(a\ g)}) \setminus \{g\} = \{g, b\} \setminus \{g\} = \{b\}$ and $\mathtt{supp}((b\ e)) = \{b, e\}$.

However, it is not possible to derive $(a\,e) \curlywedge X \vdash (a\,b) \curlywedge [a](\mathsf{f}a, (c\,d) \cdot X)$ since in this case $b$ is not in the support of the permutations that fix $X$ (only $(a\ e)$ fixes $X$ according to our context).

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

**Lemma 3.6** (Inversion)**.** *The inference rules for $\overset{\curlywedge}{\approx}_\alpha$ are invertible.*

*Proof.* Straightforward since there is only one rule for each syntactic class of terms. Note that in the case of rule ($\overset{\curlywedge}{\approx}_\alpha \mathbf{ab}$), the permutative convention ensures that $a$ and $b$ are different atoms. □

Equivariance is an important notion in nominal logic: equivariant subsets of nominal sets are closed under permutation (see Definition 1.8 in [Pit13]). Below we show that the fixed-point and $\alpha$-equivalence relations are equivariant.

**Lemma 3.7** (Equivariance)**.**

(1) $\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge\ t$ *iff* $\Upsilon \vdash \mathit{И}\overline{c}.\pi^\rho \curlywedge \rho \cdot t$*, for any permutation* $\rho$*.*

(2) $\Upsilon \vdash \mathit{И}\overline{c}.s \overset{\curlywedge}{\approx}_\alpha t$ *iff* $\Upsilon \vdash \mathit{И}\overline{c}.\pi \cdot s \overset{\curlywedge}{\approx}_\alpha \pi \cdot t$*, for any permutation* $\pi$*.*

*Proof.* For both parts the "if" statement is trivial, by taking the identity permutation, *Id*. We prove the "only if" statements.

(1) The proof is by rule induction using the rules in Figure 3. We consider cases depending on the last rule applied in the derivation.

(a) The rule applied is ($\curlywedge\mathbf{a}$).
In this case, $t = a$. Notice that, $\Upsilon \vdash \pi' \curlywedge a$ if, and only if, $\pi'(a) = a$. We want to show that $\Upsilon \vdash \mathit{И}\overline{c}.\pi^\rho \curlywedge \rho(a)$, for any permutation $\rho$. Notice that $\pi^\rho(\rho(a)) = (\rho \circ \pi \circ \rho^{-1})(\rho(a)) = (\rho \circ \pi)(a) = \rho(\pi(a)) = \rho(a)$, and the result follows.

(b) The rule applied is ($\curlywedge\mathbf{var}$).
In this case one has $t = \pi_1 \cdot X$ and $\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge \pi_1 \cdot X$. We need to prove that $\Upsilon \vdash \mathit{И}\overline{c}.\pi^\rho \curlywedge \rho \cdot (\pi_1 \cdot X)$, which requires $\mathtt{supp}((\rho \circ \pi_1)^{-1} \circ \pi^\rho \circ (\rho \circ \pi_1)) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$.
Note that, since $\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge \pi_1 \cdot X$, it follows that $\mathtt{supp}(\pi_1^{-1} \circ \pi \circ \pi_1) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$. By definition of $\pi^\rho$, it follows that

$$\begin{aligned}\mathtt{supp}((\rho \circ \pi_1)^{-1} \circ \pi^\rho \circ (\rho \circ \pi_1)) &= \mathtt{supp}((\rho \circ \pi_1)^{-1} \circ (\rho \circ \pi \circ \rho^{-1}) \circ (\rho \circ \pi_1)) \\ &= \mathtt{supp}((\pi_1^{-1} \circ \rho^{-1})) \circ (\rho \circ \pi \circ \rho^{-1}) \circ (\rho \circ \pi_1) \\ &= \mathtt{supp}(\pi_1^{-1} \circ \pi \circ \pi_1)\end{aligned}$$

Therefore, the result follows.

(c) The rule applied is ($\curlywedge$**abs**).
In this case, $t = [a]t'$ and $\Upsilon \vdash И\overline{c}.\pi \curlywedge [a]t'$, therefore, there exists a proof $\Pi$ :

$$\dfrac{\dfrac{\Pi}{\Upsilon \vdash И\overline{c}, c_1.\ \pi \curlywedge (a\ c_1)\cdot t'}}{\Upsilon \vdash И\overline{c}.\ \pi \curlywedge [a]t'}\ (\curlywedge\mathbf{abs})$$

By induction hypothesis, for an arbitrary permutation $\rho$, there exists a derivation $\Pi'$ for :

$$\Upsilon \vdash И\overline{c}, c_1.\ \pi^{\rho} \curlywedge \rho\cdot((a\ c_1)\cdot t').$$

Since $\rho\cdot((a\ c_1)\cdot t') = (\rho(a)\ \rho(c_1))\cdot(\rho\cdot t') = (\rho(a)\ c_1)\cdot(\rho\cdot t')$ because $c_1$ is a new name, we have

$$\dfrac{\dfrac{\Pi'}{\Upsilon \vdash И\overline{c}, c_1.\ \pi^{\rho} \curlywedge (\rho(a)\ c_1)\cdot(\rho\cdot t')}}{\Upsilon \vdash И\overline{c}.\ \pi^{\rho} \curlywedge [\rho(a)]\rho\cdot t'}$$

and the result follows.

(d) The rule applied is ($\curlywedge$f) or ($\curlywedge$**tuple**).
These cases follow directly by induction hypothesis.

(2) The proof is by rule induction using the rules in Figure 4.

(a) The rule applied is ($\overset{\curlywedge}{\approx}_{\alpha}$ **a**).
In this case, $s \overset{\curlywedge}{\approx}_{\alpha} t$ is an instance of $a \overset{\curlywedge}{\approx}_{\alpha} a$. It is straightforward to check that $\Upsilon \vdash И\overline{c}.\pi\cdot a \overset{\curlywedge}{\approx}_{\alpha} \pi\cdot a$, via an application of the rule ($\overset{\curlywedge}{\approx}_{\alpha}$ **a**).

(b) The rule applied is ($\overset{\curlywedge}{\approx}_{\alpha}$ **var**).
In this case, $\Upsilon \vdash И\overline{c}.\ \pi_1\cdot X \overset{\curlywedge}{\approx}_{\alpha} \pi_2\cdot X$ and $\mathtt{supp}(\pi_2^{-1}\circ\pi_1)\setminus\{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$. Notice that $\mathtt{supp}((\pi\circ\pi_2)^{-1}\circ\pi\circ\pi_1) = \mathtt{supp}(\pi_2^{-1}\circ\pi_1)$. Therefore, by rule ($\overset{\curlywedge}{\approx}_{\alpha}$ **var**), $\Upsilon \vdash И\overline{c}.\ \pi\cdot(\pi_1\cdot X) \overset{\curlywedge}{\approx}_{\alpha} \pi\cdot(\pi_2\cdot X)$.

(c) The cases corresponding to rule ($\overset{\curlywedge}{\approx}_{\alpha}$ **f**) and rule ($\overset{\curlywedge}{\approx}_{\alpha}$ [**a**]) follow directly by induction hypothesis.

(d) The rule applied is ($\overset{\curlywedge}{\approx}_{\alpha}$ **ab**).
In this case, there exist derivations $\Pi_1$ and $\Pi_2$ such that

$$\dfrac{\dfrac{\Pi_1}{\Upsilon \vdash И\overline{c}.\ t \overset{\curlywedge}{\approx}_{\alpha} (a\ b)\cdot s} \qquad \dfrac{\Pi_2}{\Upsilon \vdash И\overline{c}, c_1.\ (a\ c_1) \curlywedge s}}{\Upsilon \vdash И\overline{c}.\ [a]t \overset{\curlywedge}{\approx}_{\alpha} [b]s}\ (\overset{\curlywedge}{\approx}_{\alpha}\mathbf{ab})$$

By induction hypothesis, and part (1) of this lemma:

(i) there exists a proof $\Pi_1'$ of $\Upsilon \vdash И\overline{c}.\ \pi\cdot t \overset{\curlywedge}{\approx}_{\alpha} \pi\cdot((a\ b)\cdot s)$.
(ii) there exists a proof $\Pi_2'$ of $\Upsilon \vdash И\overline{c}, c_1.\ (a\ c_1)^{\pi} \curlywedge \pi\cdot s$.

It is easy to show that $\pi\cdot((a\ b)\cdot s) = (\pi(a)\ \pi(b))\cdot(\pi\cdot s)$, and since $c_1$ is new, $\pi(c_1) = c_1$, therefore, the result follows.

$$\dfrac{\dfrac{\Pi_1'}{\Upsilon \vdash И\overline{c}.\ \pi\cdot t \overset{\curlywedge}{\approx}_{\alpha} (\pi(a)\ \pi(b))\cdot\pi\cdot s} \qquad \dfrac{\Pi_2'}{\Upsilon \vdash И\overline{c}, c_1.\ (\pi(a)\ c_1) \curlywedge \pi\cdot s}}{\Upsilon \vdash И\overline{c}.\ [\pi(a)]\pi\cdot t \overset{\curlywedge}{\approx}_{\alpha} [\pi(b)]\pi\cdot s}\ (\overset{\curlywedge}{\approx}_{\alpha}\mathbf{ab})$$

□

**Example 3.8.** Notice that $(a\ c) \curlywedge X \vdash (a\ b) \curlywedge (b\ c) \cdot X$, for $(a\ c) \curlywedge X \vdash (a\ c) \curlywedge X$ and $(a\ b)^{(b\ c)} = (a\ c)$. Using the Equivariance Lemma, with $\rho = (b\ c)$, we deduce:

$$(a\ c) \curlywedge X \vdash (a\ b)^{(b\ c)} \curlywedge X \Leftrightarrow (a\ c) \curlywedge X \vdash (a\ b) \curlywedge (b\ c) \cdot X \tag{3.1}$$

**Proposition 3.9** (Strengthening for $\curlywedge$)**.**
*If* $\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge s$ *and* $\mathit{supp}(\pi) \subseteq \mathit{supp}(\mathit{perm}(\Upsilon|_X))$ *or* $X \notin \mathit{Var}(s)$ *then* $\Upsilon \vdash И\overline{c}.\pi' \curlywedge s$.

*Proof.* By induction on the structure of $s$.

- $s = a$
  In this case, $\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge a$. By rule ($\curlywedge\mathbf{a}$),

$$\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge a \Leftrightarrow \pi'(a) = a \Leftrightarrow \Upsilon \vdash И\overline{c}.\ \pi' \curlywedge a$$

- $s = \pi_1 \cdot X$ (the case $s = \pi_1 \cdot Y$ is trivial).
  In this case, $\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge \pi_1 \cdot X$.

$$\begin{aligned}
\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge \pi_1 \cdot X &\Leftrightarrow \mathtt{supp}((\pi')^{\pi_1^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X) \cup \{\pi\}) \quad (\textit{rule } (\curlywedge\mathbf{var})) \\
&\Leftrightarrow \mathtt{supp}((\pi')^{\pi_1^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X)) \qquad (\textit{by assumption}) \\
&\Leftrightarrow \Upsilon \vdash И\overline{c}.\pi' \curlywedge \pi_1 \cdot X \qquad (\textit{by rule } (\curlywedge\mathbf{var}))
\end{aligned}$$

- $s = [a]s'$
  In this case, $\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge [a]s'$, $\mathtt{supp}(\pi) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ or $X \notin \mathtt{Var}(s')$. Then there exists a derivation

$$\dfrac{\dfrac{\mathcal{D}}{\Upsilon, \pi \curlywedge X \vdash И\overline{c}, c_1.\ \pi' \curlywedge (a\ c_1) \cdot s'}}{\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ \pi' \curlywedge [a]s'}\ (\curlywedge\mathbf{abs})$$

  By induction hypothesis, there exists a derivation $\mathcal{D}'$ such that

$$\dfrac{\dfrac{\mathcal{D}'}{\Upsilon \vdash И\overline{c}, c_1.\ \pi' \curlywedge (a\ c_1) \cdot s'}}{\Upsilon \vdash \pi' \curlywedge [a]s'}\ (\curlywedge\mathbf{abs})$$

- $s = \mathtt{f}s'$ or $s = (s_1, \ldots s_n)$
  These cases follow easily by induction hypothesis. □

**Proposition 3.10** (Strengthening for $\overset{\curlywedge}{\approx}_\alpha$)**.**
*If* $\Upsilon, \pi \curlywedge X \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t$ *and* $\mathit{supp}(\pi) \subseteq \mathit{supp}(\mathit{perm}(\Upsilon|_X))$ *or* $X \notin \mathit{Var}(s,t)$, *then* $\Upsilon \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t$.

*Proof.* The proof is similar to that of Proposition 3.9 and omitted. □

The following auxiliary lemma permits to deduce a fixed-point constraint for a permutation $\pi$ and term $t$ from constraints involving $t$. Intuitively, it states that we can deduce that $\pi$ fixes $t$ if the support of $\pi$ is contained in permutations that fix $t$.

**Lemma 3.11.** *Let* $I$ *be a set of indices. If* $\Upsilon \vdash И\overline{c_i}.\pi_i \curlywedge t$ *for all* $i \in I$, *and* $\pi$ *is a permutation such that* $\mathit{supp}(\pi) \setminus \{\overline{c}\} \subseteq \bigcup_{i\in I} \mathit{supp}(\pi_i) \setminus \bigcup_{i\in I} \overline{c_i}$ *then* $\Upsilon \vdash И\overline{c}.\pi \curlywedge t$.

*Proof.* By induction on $t$.

- If $t = a$ then by assumption $\pi_i(a) = a$ for all $i \in I$ since $\Upsilon \vdash И\overline{c_i}.\pi_i \curlywedge a$ (rule ($\curlywedge$**a**)). Then, $\pi(a) = a$ by the assumption on the support of $\pi$. The result follows by rule ($\curlywedge$**a**).
- If $t = \pi' \cdot X$ then $\mathtt{supp}(\pi_i^{\pi'^{-1}}) \setminus \{\overline{c_i}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ because $\Upsilon \vdash И\overline{c_i}.\pi_i \curlywedge t$ for all $i \in I$ (rule ($\curlywedge$**var**)).

  Since $\mathtt{supp}(\pi)\setminus\{\overline{c}\} \subseteq \bigcup_{i\in I} \mathtt{supp}(\pi_i)\setminus\bigcup_{i\in I} \overline{c_i}$, also $\mathtt{supp}(\pi^{\pi'^{-1}})\setminus\{\overline{c}\} \subseteq \bigcup_{i\in I} \mathtt{supp}(\pi_i^{\pi'^{-1}})\setminus \bigcup_{i\in I} \overline{c_i}$. The result follows by rule ($\curlywedge$**var**).
- The other cases follow directly by induction. □

The following correctness property states that $\curlywedge$ is indeed the fixed-point relation.

**Theorem 3.12** (Correctness)**.** *Let $\Upsilon$, $\pi$ and $t$ be a fixed-point context, a permutation and a nominal term, respectively. $\Upsilon \vdash И\overline{c}.\ \pi \curlywedge t$ iff $\Upsilon \vdash И\overline{c}.\ \pi \cdot t \mathrel{\overset{\curlywedge}{\approx}_\alpha} t$.*

*Proof.* In both directions the proof is by induction on the structure of the term $t$, distinguishing cases according to the last rule applied in the derivation.
($\Rightarrow$) The interesting cases correspond to variables and abstractions; the other cases follow directly by the induction hypothesis.

(1) The last rule is ($\curlywedge$**var**).

In this case, $t = \pi' \cdot X$ and $\mathtt{supp}((\pi')^{-1} \circ \pi \circ \pi') \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$, and therefore $\Upsilon \vdash И\overline{c}.\ \pi \cdot (\pi' \cdot X) \mathrel{\overset{\curlywedge}{\approx}_\alpha} \pi' \cdot X$, via rule ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}$ **var**).

(2) The last rule is ($\curlywedge$**abs**). In this case, $t = [a]t'$ and by assumption there is a derivation of the form:

$$\dfrac{\dfrac{\Pi}{\Upsilon \vdash И\overline{c}, c_1.\ \pi \curlywedge (a\ c_1) \cdot t'}}{\Upsilon \vdash И\overline{c}.\pi \curlywedge [a]t'}$$

We need to prove that $\Upsilon \vdash И\overline{c}.\ [\pi(a)]\pi \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} [a]t'$, that is, $\Upsilon \vdash И\overline{c}.\ \pi \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} (\pi(a)\ a) \cdot t'$ and also $\Upsilon \vdash И\overline{c}, c_1.(\pi(a)\ c_1) \curlywedge t'$.

By IH, there exists a proof $\Pi'$ for $\Upsilon \vdash И\overline{c}, c_1.\pi \cdot ((a\ c_1).t') \mathrel{\overset{\curlywedge}{\approx}_\alpha} (a\ c_1) \cdot t'$. The following equivalence holds, since $c_1$ is newly quantified:

$$\Upsilon \vdash И\overline{c}, c_1.\pi \cdot ((a\ c_1) \cdot t') \mathrel{\overset{\curlywedge}{\approx}_\alpha} (a\ c_1) \cdot t' \iff \Upsilon \vdash И\overline{c}, c_1.(\pi(a)\ c_1) \cdot (\pi \cdot t') \mathrel{\overset{\curlywedge}{\approx}_\alpha} (a\ c_1) \cdot t' \tag{3.2}$$

Also, $\Upsilon \vdash И\overline{c}, c_1.(\pi \cdot t') \mathrel{\overset{\curlywedge}{\approx}_\alpha} (\pi(a)\ c_1) \cdot ((a\ c_1) \cdot t')$ by Equivariance. Note that $(\pi(a)\ c_1) \cdot ((a\ c_1) \cdot t') = (\pi(a)\ a) \cdot t'$, hence $\Upsilon \vdash И\overline{c}.\pi \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} (\pi(a)\ a) \cdot t'$ as required.

Since $\Upsilon \vdash И\overline{c}, c_1.\ \pi \curlywedge (a\ c_1) \cdot t'$, we also have $\Upsilon \vdash И\overline{c}, c_1.\ \pi^{(a\ c_1)} \curlywedge t'$ by equivariance and we deduce $\Upsilon \vdash И\overline{c}, c_1.(\pi(a)\ c_1) \curlywedge t'$ as required.

($\Leftarrow$) Similarly, we proceed by induction on the derivation of $\Upsilon \vdash И\overline{c}.\ \pi \cdot t \mathrel{\overset{\curlywedge}{\approx}_\alpha} t$. We show the interesting cases, the others follow directly by induction.

(1) The rule is ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}$ **var**).

In this case $t = \pi_1 \cdot X$ and there exists a proof of $\Upsilon \vdash И\overline{c}.\ \pi \cdot (\pi_1 \cdot X) \mathrel{\overset{\curlywedge}{\approx}_\alpha} \pi_1 \cdot X$. Therefore, $\mathtt{supp}(\pi_1^{-1} \circ \pi \circ \pi_1) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$, and one can conclude that $\Upsilon \vdash И\overline{c}.\ \pi \curlywedge \pi_1 \cdot X$, via application of rule ($\curlywedge$**var**).

(2) The rule is ($\mathrel{\overset{\curlywedge}{\approx}_\alpha}$ **ab**).

In this case, $t = [a]t'$ and since $\Upsilon \vdash И\overline{c}.\ [\pi(a)]\pi \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} [a]t'$, we know (Inversion): $\Upsilon \vdash И\overline{c}.\ \pi \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} (\pi(a)\ a) \cdot t'$ and $\Upsilon \vdash И\overline{c}, c_1.(\pi(a)\ c_1) \curlywedge t'$.

By Equivariance $\Upsilon \vdash И\overline{c}.((\pi(a)\ a) \circ \pi) \cdot t' \mathrel{\overset{\curlywedge}{\approx}_\alpha} t'$ and by induction $\Upsilon \vdash И\overline{c}.((\pi(a)\ a) \circ \pi) \curlywedge t'$.

From $\Upsilon \vdash И\overline{c}, c_1.(\pi(a)\ c_1) \curlywedge t'$ and $\Upsilon \vdash И\overline{c}.((\pi(a)\ a) \circ \pi) \curlywedge t'$ we deduce $\Upsilon \vdash И\overline{c}, c_1.\pi^{(a\ c_1)} \curlywedge t'$ by Lemma 3.11, since $\mathtt{supp}(\pi^{(a\ c_1)}) \setminus \{\overline{c}, c_1\} \subseteq (\mathtt{supp}((\pi(a)\ a) \circ \pi) \setminus \{\overline{c}\}) \cup (\mathtt{supp}((\pi(a)\ c_1)) \setminus \{c_1\})$. By Equivariance, $\Upsilon \vdash И\overline{c}, c_1.\pi \curlywedge (a\ c_1) \cdot t'$, and therefore $\Upsilon \vdash И\overline{c}.\ \pi \curlywedge [a]t'$ by $(\curlywedge\mathbf{ab})$, as required. □

Recall that $\Upsilon\sigma$ denotes the set of fixed-point constraints obtained by applying the substitution $\sigma$ to the constraints in $\Upsilon$ (see Section 3.1). Below we abbreviate $\Upsilon \vdash \pi_1 \curlywedge t_1, \ldots, \Upsilon \vdash \pi_n \curlywedge t_n$ as $\Upsilon \vdash \pi_1 \curlywedge t_1, \ldots, \pi_n \curlywedge t_n$. Thus, $\Upsilon \vdash \Upsilon'\sigma$ means that each of the constraints in $\Upsilon'\sigma$ is derivable from $\Upsilon$.

**Proposition 3.13** (Preservation under Substitution)**.** *Suppose that* $\Upsilon \vdash \Upsilon'\sigma$*. Then,*

(1) $\Upsilon' \vdash И\overline{c}.\ \pi \curlywedge s \Longrightarrow \Upsilon \vdash И\overline{c}.\ \pi \curlywedge s\sigma$.

(2) $\Upsilon' \vdash И\overline{c}.\ s \mathrel{\overset{\curlywedge}{\approx}_\alpha} t \Longrightarrow \Upsilon \vdash И\overline{c}.\ s\sigma \mathrel{\overset{\curlywedge}{\approx}_\alpha} t\sigma$.

*Proof.* By induction on the rules in Figures 3 and 4.

(1) We distinguish cases depending on the last rule applied in the derivation of $\Upsilon' \vdash И\overline{c}.\pi \curlywedge s$.

(a) The rule is $(\curlywedge\mathbf{a})$.
In this case, $s = a$ and
$$\frac{\pi(a) = a}{\Upsilon' \vdash И\overline{c}.\pi \curlywedge a}\ (\curlywedge\mathbf{a})$$
The result follows trivially, since $a\sigma = a$ for any substitution $\sigma$.

(b) The rule is $(\curlywedge\mathbf{f})$
In this case, $s = \mathsf{f}s'$ and there exists a proof $\Pi'$ of
$$\frac{\dfrac{\Pi'}{\Upsilon' \vdash И\overline{c}.\pi \curlywedge s'}}{\Upsilon' \vdash И\overline{c}.\pi \curlywedge \mathsf{f}s'}\ (\curlywedge\mathbf{f})$$
By induction hypothesis, there exists a proof $\Pi''$ such that
$$\frac{\dfrac{\Pi''}{\Upsilon \vdash И\overline{c}.\pi \curlywedge s'\sigma}}{\Upsilon \vdash И\overline{c}.\pi \curlywedge (\mathsf{f}s')\sigma}\ (\curlywedge\mathbf{f})$$

(c) The rule is $(\curlywedge\mathbf{abs})$
In this case $s = [a]s'$ and there exists a proof $\Pi'$ of the form
$$\frac{\dfrac{\Pi'}{\Upsilon' \vdash И\overline{c}, c_1.\pi \curlywedge (a\ c_1) \cdot s'}}{\Upsilon' \vdash И\overline{c}.\pi \curlywedge [a]s'}\ (\curlywedge\mathbf{abs})$$
By induction hypothesis, there exists a proof $\Pi''$ of the form
$$\frac{\dfrac{\Pi}{\Upsilon \vdash И\overline{c}, c_1.\pi \curlywedge ((a\ c_1) \cdot s')\sigma}}{}$$
Since $((a\ c_1) \cdot s')\sigma = (a\ c_1) \cdot (s'\sigma)$ and $[a](s'\sigma) = ([a]s')\sigma$, it follows that
$$\frac{\dfrac{\Pi}{\Upsilon \vdash И\overline{c}, c_1.\pi \curlywedge (a\ c_1) \cdot (s'\sigma)}}{\Upsilon \vdash И\overline{c}.\pi \curlywedge [a](s'\sigma)}\ (\curlywedge\mathbf{abs})$$

(d) The rule is $(\curlywedge\mathbf{tuple})$
This case is analogous to the case for $(\curlywedge\mathbf{f})$, and follows directly by IH.

(e) The rule is ($\curlywedge$**var**)
In this case, $s = \rho \cdot X$ and $\Upsilon' \vdash И\overline{c}.\pi \curlywedge \rho \cdot X$ holds, that is, $\mathtt{supp}(\pi^{\rho^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon'|_X))$.
From $\Upsilon \vdash \Upsilon'\sigma$, it follows that, $\Upsilon \vdash \pi_1 \curlywedge X\sigma$, for all $\pi_1 \curlywedge X \in \Upsilon'$.
Therefore, $\Upsilon \vdash И\overline{c}.\ \pi^{\rho^{-1}} \curlywedge X\sigma$ by Lemma 3.11, and the result follows by Equivariance.

(2) We proceed by analysing the last rule used in the derivation of $\Upsilon' \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t$.

(a) The last rule is ($\overset{\curlywedge}{\approx}_\alpha$ **a**).
This case is trivial.

(b) The last rule is ($\overset{\curlywedge}{\approx}_\alpha$ **var**).
In this case we have $\Upsilon' \vdash И\overline{c}.\ \pi \cdot X \overset{\curlywedge}{\approx}_\alpha \pi' \cdot X$ and therefore $\mathtt{supp}((\pi')^{-1} \circ \pi) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon'|_X))$.
From $\Upsilon \vdash \Upsilon'\sigma$, it follows that $\Upsilon \vdash \pi_1 \curlywedge X\sigma$, for all $\pi_1 \curlywedge X \in \Upsilon'$.
Therefore, $\Upsilon \vdash И\overline{c}.(\pi')^{-1} \circ \pi \curlywedge X\sigma$ by Lemma 3.11, and the result follows by Theorem 3.12 and Equivariance.

(c) The last rule is ($\overset{\curlywedge}{\approx}_\alpha$ **f**) or ($\overset{\curlywedge}{\approx}_\alpha$ [**a**]). These cases follow directly by induction.

(d) The last rule is ($\overset{\curlywedge}{\approx}_\alpha$ **ab**).
In this case, we know $\Upsilon \vdash И\overline{c}.\ [a]s \overset{\curlywedge}{\approx}_\alpha [b]t$ and therefore (by Inversion) $\Upsilon \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha (a\ b) \cdot t$ and $\Upsilon \vdash И\overline{c}, c_1.(a\ c_1) \curlywedge t$.
By induction hypothesis, $\Upsilon \vdash И\overline{c}.\ s\sigma \overset{\curlywedge}{\approx}_\alpha (a\ b) \cdot t\sigma$, and by part (1) of this proposition, $\Upsilon \vdash И\overline{c}, c_1.(a\ c_1) \curlywedge t\sigma$. The result then follows using rule ($\overset{\curlywedge}{\approx}_\alpha$ **ab**). □

**Corollary 3.14** (Weakening)**.** *Assume* $\Upsilon' \subseteq \Upsilon$.
(1) $\Upsilon' \vdash И\overline{c}.\ \pi \curlywedge s \Longrightarrow \Upsilon \vdash И\overline{c}.\ \pi \curlywedge s$.
(2) $\Upsilon' \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t \Longrightarrow \Upsilon \vdash И\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t$.

3.2. **Alternative approaches to new name generation.** In the previous section we used the "new" quantifier to deal with new names in constraints: judgements involve newly quantified constraints, and in rules ($\curlywedge$**ab**) and ($\overset{\curlywedge}{\approx}_\alpha$ **ab**), when new fresh names are needed, a newly quantified atom is used.

An alternative approach consists of quantifying judgements instead of constraints. More precisely, we can define judgements of the form $И\overline{c}.(\Upsilon \vdash \pi \curlywedge t)$ and $И\overline{c}.(\Upsilon \vdash s \overset{\curlywedge}{\approx}_\alpha t)$. Rules to derive this kind of judgements can be easily obtained by adapting the rules in Figures 3 and 4. We show the rules for fixed-point judgements in Figure 5. For the sake of readabilty, we omit brackets around judgements in the rules and assume the quantifiers have maximal scope.

Yet another approach consists of using a name generator whenever new names are required (see [CU08, CM17]), as done in [AFN18]. In this case, the new quantifier is not needed, and new names are generated dynamically by using an external generator, under the assumption that the generator outputs a new (unused name) whenever needed. For comparison, we recall in Figure 6 the rules given in [AFN18] to derive fixed-point judgements $\Upsilon \vdash \pi \curlywedge t$. Note that in rule ($\curlywedge$**abs**) the fixed-point context $\Upsilon$ is augmented with constraints $(c_1\ c_2) \curlywedge \mathtt{Var}(t)$. These constraints serve to store the information about the fact that the generated atoms are "new". The trick is to generate two atoms, even though only one new

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

$$\frac{\mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge t}{\mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge \mathtt{f}\ t}\,(\curlywedge\mathtt{f}) \qquad \frac{\mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge t_1 \quad \ldots \quad \mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge t_n}{\mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge (t_1, \ldots, t_n)}\,(\curlywedge\mathbf{tuple})$$

$$\frac{\mathit{И}\bar{c}, c_1.\Upsilon \vdash \pi \curlywedge (a\ c_1) \cdot t}{\mathit{И}\bar{c}.\Upsilon \vdash \pi \curlywedge [a]t}\,(\curlywedge\mathbf{abs})$$

Figure 5: Alternative fixed-point rules.

atom is needed ($c_1$, to replace $a$), in order to be able to express the fact that $c_1$ is new. This solution is inspired by the link between freshness, the new quantifier and fixed-point equations, $a\#X \Leftrightarrow \mathit{И}a'.(a\ a') \cdot X = X$, mentioned at the beginning of Section 3.

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

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

$$\frac{\Upsilon, (c_1\ c_2) \curlywedge \mathtt{Var}(t) \vdash \pi \curlywedge (a\ c_1) \cdot t}{\Upsilon \vdash \pi \curlywedge [a]t}\,(\curlywedge\mathbf{abs})\ \textit{where } c_1 \textit{ and } c_2 \textit{ are new names}$$

Figure 6: Fixed-point rules using a name generator.

The rules to derive $\alpha$-equivalence judgements following this approach are recalled in Figure 7.

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

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

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

Figure 7: Equality rules using a name generator. In rule $(\overset{\curlywedge}{\approx}_\alpha\mathbf{ab})$, $c_1$ and $c_2$ are new names.

The latter approach has the advantage of having a simpler syntax for judgements (without quantification) but relies on an external name generator, which can be seen as adding a form of state. In other words, in the latter approach rules produce side-effects. Although not as elegant as the approaches using $\mathit{И}$, this approach is convenient from an implementation point of view. In the next sections we use this technique to translate primitive freshness constraints to primitive fixed-point constraints, and to specify a unification algorithm.

3.3. **From freshness to fixed-point constraints and back again.** In this section we show that the $\alpha$-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 primitive freshness to primitive fixed-point constraints and a transformation $[\_]^{\#}$ from primitive fixed-point to primitive freshness constraints can be defined.

Below we denote by $\mathfrak{F}_\#$ the family of freshness contexts, and by $\mathfrak{F}_\curlywedge$ the family of fixed-point contexts. The mapping $[\_]_\curlywedge$ associates each primitive freshness constraint with a fixed-point constraint; it extends to freshness contexts in the natural way.

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

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

The mapping $[\_]_\#$ associates each primitive fixed-point constraint with a freshness constraint; it extends to fixed-point contexts in the natural way.

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

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

Below we abbreviate the set of constraints $\{a_1\#t, \ldots a_n\#t \mid a_1, \ldots, a_n \in A\}$ as $\overline{A\#t}$.

Using the above-specified translation of primitive constraints, we translate freshness judgements into fixed-point judgements with newly quantified constraints:

$$\Delta \vdash a\#t \text{ is translated as } [\Delta]_\curlywedge \vdash \curlyvee c.(a\ c) \curlywedge t$$

and vice-versa:

$$\Upsilon \vdash \curlyvee \bar{c}.\pi \curlywedge t \text{ is translated as } [\Upsilon]_\# \vdash \overline{\mathtt{supp}(\pi) \setminus \{\bar{c}\}\#t}.$$

**Theorem 3.15.**

(1) $\Delta \vdash a\#t \Leftrightarrow [\Delta]_\curlywedge \vdash \curlyvee c.(a\ c) \curlywedge t.$

(2) $\Upsilon \vdash \curlyvee \bar{c}.\pi \curlywedge t \Leftrightarrow [\Upsilon]_\# \vdash \overline{\mathit{supp}(\pi) \setminus \{\bar{c}\}\#t}.$

*Proof.* Part (1):

($\Rightarrow$) By induction on the derivation of $\Delta \vdash a\#t$ (see the rules in Figure 1). We distinguish cases based on the last rule used in the derivation.

- If the rule is (#**a**) then $t$ is an atom $b$ (it cannot be $a$), and the result follows by rule ($\curlywedge$**a**).
- If the rule is (#**var**) then $t = \pi \cdot X$ and since $\Delta \vdash a\#\pi \cdot X$ we know $\pi^{-1}(a)\#X \in \Delta$ by Inversion. Hence, $(\pi^{-1}(a)\ c_{\pi^{-1}(a)}) \curlywedge X \in [\Delta]_\curlywedge$, and therefore $\mathtt{supp}((a\ c)^{\pi^{-1}}) \setminus \{c\} \subseteq \mathtt{supp}(\mathtt{perm}([\Delta]_\curlywedge|_X))$ (recall that $\pi^{-1}(c) = c$ since $c$ is a new atom). The result then follows by rule ($\curlywedge$**var**).
- The cases for (#**tuple**) and (#**f**) follow directly by induction.
- If the rule is (#[**a**]) then $t = [a]s$ and we need to prove $[\Delta]_\curlywedge \vdash \curlyvee c.(a\ c) \curlywedge [a]s$. By rule ($\curlywedge$**abs**), it suffices to show $[\Delta]_\curlywedge \vdash \curlyvee c, c_1.(a\ c) \curlywedge (a\ c_1) \cdot s$. By Equivariance, this is equivalent to $[\Delta]_\curlywedge \vdash \curlyvee c, c_1.(a\ c)^{(a\ c_1)} \curlywedge s$, or equivalently $[\Delta]_\curlywedge \vdash \curlyvee c, c_1.(c_1\ c) \curlywedge s$, which holds trivially since $c$ and $c_1$ are new atoms ($(c_1\ c) \cdot s = s$)
- If the rule is (#**abs**) then $t = [b]s$. By assumption, $\Delta \vdash a\#[b]s$, hence $\Delta \vdash a\#s$. By induction hypothesis, $[\Delta]_\curlywedge \vdash \curlyvee c.(a\ c) \curlywedge s$, and by Equivariance, $[\Delta]_\curlywedge \vdash \curlyvee c, c_1.(a\ c) \curlywedge (b\ c_1) \cdot s$. The result follows by rule ($\curlywedge$**abs**).

($\Leftarrow$) By induction on the derivation of $[\Delta]_\curlywedge \vdash \curlyvee c.(a\ c) \curlywedge t$. We distinguish cases based on the last rule used in the derivation.

- If the rule is ($\curlywedge$**a**) then $t$ is an atom $b$ (it cannot be $a$), and the result follows by rule (#**a**).
- If the rule is ($\curlywedge$**var**) then $t = \pi \cdot X$ and since $[\Delta]_{\curlywedge} \vdash \mathit{И}c.(a\ c) \curlywedge \pi \cdot X$ we know $\mathtt{supp}((a\ c)^{\pi^{-1}}) \setminus \{c\} \subseteq \mathtt{supp}(\mathtt{perm}([\Delta]_{\curlywedge}|_X))$ by Inversion. Hence, $\pi^{-1}(a) \in \mathtt{supp}(\mathtt{perm}([\Delta]_{\curlywedge}|_X))$, and therefore $\pi^{-1}(a)\#X \in \Delta$ by definition of the mapping $[\cdot]_{\curlywedge}$. The result then follows by rule (#**var**).
- The cases for ($\curlywedge$**tuple**) and ($\curlywedge$**f**) follow directly by induction.
- If the rule is ($\curlywedge$**abs**) then there are two cases, $t = [a]s$ and $t = [b]s$.
  If $t = [a]s$ the result follows directly by rule (#[**a**]).
  If $t = [b]s$ then by assumption and Inversion, $[\Delta]_{\curlywedge} \vdash \mathit{И}cc_1.(a\ c) \curlywedge (b\ c_1) \cdot s$. By Equivariance, $[\Delta]_{\curlywedge} \vdash \mathit{И}cc_1.(a\ c) \curlywedge s$. By induction hypothesis we deduce $\Delta \vdash a\#s$, and the result follows by rule (#**abs**).

Part (2):

($\Rightarrow$) By induction on the derivation of $\Upsilon \vdash \mathit{И}\overline{c}.\pi \curlywedge t$. Again we distinguish cases based on the last rule used in the derivation. The only interesting cases are rule ($\curlywedge$**var**) and ($\curlywedge$**abs**).

- If the last rule applied is ($\curlywedge$**var**) then $t = \pi' \cdot X$. By Inversion, $\mathtt{supp}(\pi^{\pi'^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$. Therefore, for any $a \in \mathtt{supp}(\pi) \setminus \{\overline{c}\}$, $\pi'^{-1}(a) \in \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$. By definition of the mapping $[\cdot]_{\#}$, $\pi'^{-1}(a)\#X \in [\Upsilon]_{\#}$, and the result follows by rule (#**var**).
- If the last rule applied is ($\curlywedge$**abs**) then $t = [a]s$. In this case, we know $\Upsilon \vdash \mathit{И}c.c_1.\pi \curlywedge (a\ c_1) \cdot s$. By induction, $[\Upsilon]_{\#} \vdash \mathtt{supp}(\pi) \setminus \{c, c_1\}\#(a\ c_1) \cdot s$, hence $[\Upsilon]_{\#} \vdash \mathtt{supp}(\pi) \setminus \{a\}\#s$, and the result follows by rules (#[**a**]) and (#**abs**).

($\Leftarrow$) By induction on $t$.

- If $t$ is an atom $a$ then $a \notin \mathtt{supp}(\pi) \setminus \{\overline{c}\}$ (by Inversion, rule (#**a**)). The result follows by rule ($\curlywedge$**a**).
- If $t = \pi' \cdot X$, then the rule applied in the derivation is rule (#**var**). By Inversion, for any atom $a$ in $\mathtt{supp}(\pi) \setminus \{\overline{c}\}$, $\mathtt{supp}(\pi^{\pi'^{-1}}) \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))$ and the result follows by rule ($\curlywedge$**var**).
- If $t = [a]s$ then for any atom $b \in \mathtt{supp}(\pi) \setminus \{\overline{c}\}$ ($b$ different from $a$ by the permutative convention), we know $[\Upsilon]_{\#} \vdash b\#s$, since by assumption $[\Upsilon]_{\#} \vdash b\#[a]s$ and the last rule applied in the derivation must have been rule (#**abs**). In particular, for any atom $b \in \mathtt{supp}(\pi) \setminus \{\overline{c}, a, c_1\}$, $[\Upsilon]_{\#} \vdash b\#s$. Then, by induction hypothesis, $\Upsilon \vdash \mathit{И}c.c_1.\pi^{(a\ c_1)} \curlywedge s$ and the result follows by Equivariance and rule ($\curlywedge$**abs**).
- The other cases follow directly by induction. □

**Theorem 3.16.** *$\overset{\curlywedge}{\approx}_\alpha$ 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) $\Upsilon \vdash \mathit{И}\overline{c}.\ s \overset{\curlywedge}{\approx}_\alpha t \Rightarrow [\Upsilon]_{\#}, \Delta \vdash s \approx_\alpha t$, *where* $\Delta \vdash \overline{c}\#\mathtt{Var}(s, t)$.

*Proof.* (1) The first part is proved by induction on the derivation of $\Delta \vdash s \approx_\alpha t$, distinguishing cases according to the last rule applied (see Figure 2). The interesting cases correspond to ($\approx_\alpha$ **var**) and ($\approx_\alpha$ **ab**).

- If the last rule applied is ($\approx_\alpha$ **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.
- If the last rule applied is $(\approx_\alpha \mathbf{ab})$ the result follows directly by induction and Theorem 3.15.

(2) The second part is proved by induction on the derivation of $\Upsilon \vdash s \overset{\curlywedge}{\approx}_\alpha t$, distinguishing cases according to the last rule applied. Again the interesting cases correspond to variables and abstractions. The proof follows the lines of the previous part and is omitted. □

As a corollary, since $\approx_\alpha$ is an equivalence relation [UPG04], we deduce that $\overset{\curlywedge}{\approx}_\alpha$ is also an equivalence relation.

**Theorem 3.17.** *$\overset{\curlywedge}{\approx}_\alpha$ is an equivalence relation.*

**Lemma 3.18** ($\curlywedge$ preservation under $\overset{\curlywedge}{\approx}_\alpha$)**.** *If $\Upsilon \vdash s \overset{\curlywedge}{\approx}_\alpha t$ and $\Upsilon \vdash \text{И}\overline{c}.\pi \curlywedge s$ then $\Upsilon \vdash \text{И}\overline{c}.\pi \curlywedge t$.*

*Proof.* Direct consequence of Theorem 3.12, Equivariance and Transitivity. □

Having proved that $\overset{\curlywedge}{\approx}_\alpha$ is an equivalence relation, and that $\curlywedge$ is correctly defined with respect to $\overset{\curlywedge}{\approx}_\alpha$, we can use $\curlywedge$ to define the support of a non-ground term. We denote the support of a term in context $\Upsilon \vdash t$ as $\mathtt{supp}_\Upsilon(t)$. As indicated in Definition 2.5, the set of atoms in the support of an element of a nominal set can be characterised by using permutation fixed points. In the particular case of terms, the previous results justify the definition of $\mathtt{supp}_\Upsilon(t)$ using fixed-point judgements as follows.

**Definition 3.19.** Let $\Upsilon \vdash t$ be a term in context. The *support* of $t$ with respect to $\Upsilon$, $\mathtt{supp}_\Upsilon(t)$, is the smallest set $A$ of atoms such that for any permutation $\pi$,

$$(\forall a \in A, \pi(a) = a) \Rightarrow \Upsilon \vdash \pi \curlywedge t.$$

As expected, $\alpha$-equivalent terms have the same support.

**Lemma 3.20.**

(1) *If $\Upsilon \vdash s \overset{\curlywedge}{\approx}_\alpha t$ then $\mathit{supp}_\Upsilon(s) = \mathit{supp}_\Upsilon(t)$.*
(2) *$\Upsilon \vdash \text{И}\overline{c}.\pi \curlywedge t$ if and only if $\mathit{supp}(\pi) \setminus \{\overline{c}\} \cap \mathit{supp}_\Upsilon(t) = \emptyset$.*

*Proof.* Direct consequence of Definition 3.19 and Lemma 3.18. □

## 4. Nominal Unification via fixed-point constraints

In this section we address the problem of unifying nominal terms. Solutions for unification problems will be represented using fixed-point constraints and substitutions. After defining unification problems, we present a simplification algorithm that computes the most general unifier for a unification problem, provided the problem has a solution (otherwise the algorithm stops indicating that there is no solution). To specify the nominal unification algorithm, in this section we follow the approach to dealing with new atoms that relies on a name generator.

**Definition 4.1.** A *unification problem* $\mathtt{Pr}$ consists of a finite set of equality and fixed-point constraints of the form $s \overset{\curlywedge^?}{\approx}_\alpha t$ and $\pi \curlywedge^? t$, respectively.

Below we call $\pi \between^? X$ a **primitive** constraint.

**Definition 4.2** (Solution)**.** A *solution* for a problem $\mathtt{Pr}$ is a pair of the form $\langle \Phi, \sigma \rangle$ where the following conditions are satisfied:

(1) $\Phi \vdash \pi \between t\sigma$, if $\pi \between^? t \in \mathtt{Pr}$;

(2) $\Phi \vdash s\sigma \overset{\between}{\approx}_\alpha t\sigma$, if $s \overset{\between^?}{\approx}_\alpha t \in \mathtt{Pr}$.

(3) $X\sigma \overset{\between}{\approx}_\alpha X\sigma\sigma$ for all $X \in \mathtt{Var}(\mathtt{Pr})$ (the substitution is idempotent).

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

Solutions in $\mathcal{U}(\mathtt{Pr})$ can be compared using the following ordering.

**Definition 4.3.** Let $\Phi_1, \Phi_2$ be fixed-point contexts, and $\sigma_1, \sigma_2$ substitutions. Then $\langle \Phi_1, \sigma_1 \rangle \leq \langle \Phi_2, \sigma_2 \rangle$ if there exists some $\sigma'$ such that

$$\text{for all } X, \quad \Phi_2 \vdash X\sigma_1\sigma' \overset{\between}{\approx}_\alpha X\sigma_2 \quad \text{and} \quad \Phi_2 \vdash \Phi_1\sigma'.$$

**Definition 4.4.** A **principal** (or **most general**) solution to a problem $\mathtt{Pr}$ is a least element of $\mathcal{U}(\mathtt{Pr})$.

We design a unification algorithm via the simplification rules presented in Table 1. These rules act on unification problems $\mathtt{Pr}$ by transforming constraints into simpler ones, or instantiating variables in the case of rules ($\approx_\alpha$ *inst*1) and ($\approx_\alpha$ *inst*1). We call the latter *instantiating rules*. We abbreviate $(t_1, \ldots, t_n)$ as $(\widetilde{t})_{1..n}$, and for a set $S$, $\overline{\pi \between S} = \{\pi \between X \mid X \in S\}$.

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

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

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 4.5** (Termination)**.** *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 $\mathtt{Pr}$. To compare $[\mathtt{Pr}]$ and $[\mathtt{Pr}']$ we use the lexicographic combination of the usual order on natural numbers, $>$, and its multiset extension. We denote the order by $>_{lex}$. Thus, $[\mathtt{Pr}] >_{lex} [\mathtt{Pr}']$ if $\mathtt{Pr}'$ has less variables than $\mathtt{Pr}$, or if it has the same number of variables but smaller equality constraints and non-primitive fixed-point constraints.

Each simplification step $\mathtt{Pr} \Longrightarrow \mathtt{Pr}'$ 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 (when a non-instantiating rule is used). Therefore, $\mathtt{Pr} \Longrightarrow \mathtt{Pr}'$ implies $[\mathtt{Pr}] >_{lex} [\mathtt{Pr}']$. Hence, it is not possible to have an infinite descending chain. □

If $\mathtt{Pr} \Longrightarrow^* \mathtt{Pr}'$ and $\mathtt{Pr}'$ is irreducible, we say that $\mathtt{Pr}'$ is a normal form. We will show next that if instantiation rules are not used, each problem $\mathtt{Pr}$ has a unique normal form. Indeed unicity of normal forms for non-instantiating rules is a consequence of the following property.

**Lemma 4.6** (Confluence)**.** *The relation* $\Longrightarrow$ *defined by the rules in Table 1 without* $(\approx_\alpha inst1)$ *and* $(\approx_\alpha inst2)$ *is confluent.*

*Proof.* Confluence follows from the fact that the rules have no critical pairs (there are only trivial overlaps) and are terminating (by Newman's lemma). □

If instantiating rules are used, normal forms are not necessarily unique. For example, the problem $(a\ b) \curlywedge^? X, X \stackrel{\curlywedge^?}{\approx}_\alpha Y$ has two normal forms:

$$(a\ b) \curlywedge^? X, X \stackrel{\curlywedge^?}{\approx}_\alpha Y \Longrightarrow (a\ b) \curlywedge^? Y$$

$$(a\ b) \curlywedge^? X, X \stackrel{\curlywedge^?}{\approx}_\alpha Y \Longrightarrow (a\ b) \curlywedge^? X$$

However reductions do always terminate with some normal form, and all the normal forms of a problem are equivalent in a natural and useful sense (see Definition 4.9 and Theorem 4.12). We will use the notation $\langle \mathtt{Pr} \rangle_{\mathtt{nf}}$ to refer to any normal form of $\mathtt{Pr}$.

We say that an equality constraint $s \stackrel{\curlywedge^?}{\approx}_\alpha t$ 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 $t = \mathtt{g}\ t'$;
(3) $s$ and $t$ have different term constructors, that is, $s = [a]s'$ and $t = \mathtt{f}\ t'$, for some term former $\mathtt{f}$, 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 *inconsistent* whereas the latter is *consistent*.

**Example 4.7.** For $\mathtt{Pr} = [a]\mathsf{f}(X,a) \stackrel{\curlywedge^?}{\approx}_\alpha [b]\mathsf{f}((b\ c)\cdot W,(a\ c)\cdot Y)$, we obtain the following derivation chain using the rules in Table 1:

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

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

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

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

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

**Definition 4.8** (Characterisation of normal forms)**.** Let $\mathtt{Pr}$ be a problem such that $\langle \mathtt{Pr}\rangle_{\mathtt{nf}} = \mathtt{Pr}'$. We say that $\mathtt{Pr}'$ 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*.

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 $\langle \mathtt{Pr}\rangle_{\mathtt{nf}}$.

**Definition 4.9** (Computed Solutions)**.** If $\langle \mathtt{Pr}\rangle_{\mathtt{nf}}$ *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, denoted $\langle \mathtt{Pr}\rangle_{sol}$, consists of the composition $\sigma$ 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 4.10** (Continuing example 4.7)**.** Notice that $\langle \Psi,\sigma\rangle$, where $\Psi = \{(a\ c_1) \curlywedge W,(c_1\ c_2) \curlywedge W\}$ and $\sigma = \{Y \mapsto b, X \mapsto (a\ b)\circ(b\ c)\cdot W\}$, is a solution for $\mathtt{Pr}$.

We will now show that every solvable unification problem has a principal solution, computed by the unification algorithm, such that any other solution can be obtained as an instance of the principal one.

We start by proving that the non-instantiating rules preserve the set of solutions.

**Lemma 4.11** (Correctness of Non-Instantiating rules)**.** *Let $\mathtt{Pr}$ be a unification problem such that $\mathtt{Pr} \Longrightarrow^* \mathtt{Pr}'$ without using instantiating rules ($\stackrel{\curlywedge}{\approx}_\alpha$ inst1) and ($\stackrel{\curlywedge}{\approx}_\alpha$ inst2) 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 $\mathtt{Pr} \stackrel{n}{\Longrightarrow} \mathtt{Pr}'$.
**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

$$\begin{aligned}\Psi \vdash \pi \curlywedge \pi' \cdot X\sigma \quad &\Rightarrow \Psi \vdash \pi \cdot (\pi' \cdot X\sigma) \overset{\curlywedge}{\approx}_\alpha (\pi' \cdot X\sigma), \text{ hence}\\ &\Psi \vdash (\pi')^{-1} \circ \pi \circ \pi' \cdot (X\sigma) \overset{\curlywedge}{\approx}_\alpha X\sigma \text{ by Equivariance (Lemma 3.7)}\\ &\Rightarrow \Psi \vdash \pi^{(\pi')^{-1}} \curlywedge X\sigma.\end{aligned}$$

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 $\curlywedge abs$. Then

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

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 $(\curlywedge\mathbf{abs})$ (see Figure 6), this implies that there exists a proof for $\Psi, \overline{(c_1\ c_2) \curlywedge \mathtt{Var}(s\sigma)} \vdash \pi \curlywedge (a\ c_1).s\sigma$.

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 3.20, 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 3.13. The other inclusion is similar.

The cases corresponding to the other non-instantiating simplification rules are similar to the above and omitted. □

We now show that the result of the algorithm is a principal solution.

**Theorem 4.12.** *Let $\mathtt{Pr}$ be a unification problem, and suppose $\langle \mathtt{Pr}\rangle_{sol} = \langle \Phi, \sigma\rangle$. Then:*
(1) *$\langle \Phi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr})$, and*
(2) *$\langle \Phi, \sigma\rangle \leq \langle \Phi', \sigma'\rangle$ for all other $\langle \Phi', \sigma'\rangle \in \mathcal{U}(\mathtt{Pr})$. That is, the solution is also a* least *or* principal *solution.*

*Proof.* We work by induction on the length of the reduction $\mathtt{Pr} \Longrightarrow^* \langle \mathtt{Pr}\rangle_{sol}$.

- Suppose $\mathtt{Pr}$ is in normal form. Then the result is trivial since:
  (1) $\sigma = Id$, $\Phi \vdash \mathtt{Pr}Id$ and $Id$ is idempotent;
  (2) For any other $\langle \Phi', \sigma'\rangle \in \mathcal{U}(\mathtt{Pr})$, $\sigma'$ is such that $\Phi' \vdash \Phi\sigma$ and $\Phi' \vdash XId\sigma \overset{\curlywedge}{\approx}_\alpha X\sigma$ for all $X$.
- Suppose $\mathtt{Pr} \Longrightarrow \mathtt{Pr}'$ by some non-instantiating simplification. Then using Lemma 4.11, we know that $\mathcal{U}(\mathtt{Pr}) = \mathcal{U}(\mathtt{Pr}')$. Both parts of the result follow by induction.

- Suppose $\mathtt{Pr} \Longrightarrow^{\theta} \mathtt{Pr}'\theta$ by an instantiating rule. Assume $\mathtt{Pr} = \mathtt{Pr}' \cup \{\pi \cdot X \stackrel{\curlywedge}{\approx}_\alpha t\}$ where $\theta = [X \mapsto \pi^{-1} \cdot t]$ and $X \notin \mathtt{Var}(t)$ (the case for the other instantiating rule is similar). Suppose $\langle \mathtt{Pr}'\theta \rangle_{sol} = \langle \Phi, \sigma \rangle$, so that by construction $\langle \mathtt{Pr} \rangle_{sol} = \langle \Phi, \theta \circ \sigma \rangle$.
  (1) It is easy to see that $\theta \circ \sigma$ is idempotent and by the first part of the inductive hypothesis $\Phi \vdash \mathtt{Pr}'\theta\sigma$, that is, $\langle \Phi, \theta \circ \sigma \rangle \in \mathcal{U}(\mathtt{Pr})$.
  (2) Suppose $\langle \Phi', \sigma' \rangle \in \mathcal{U}(\mathtt{Pr})$. Then $\Phi' \vdash X\sigma' \stackrel{\curlywedge}{\approx}_\alpha \pi^{-1} \cdot t\sigma'$ by Equivariance. Hence, $\langle \Phi', \theta \circ \sigma'' \rangle \in \mathcal{U}(\mathtt{Pr})$ where $\sigma''$ acts just like $\sigma'$ only it maps $X$ to $X$, $\theta = [X \mapsto \pi^{-1} \cdot t]$, and $\sigma' = \theta \circ \sigma''$. Note that $\langle \Phi', \sigma'' \rangle \in \mathcal{U}(\mathtt{Pr}'\theta)$ and by inductive hypothesis $\langle \Phi, \sigma \rangle \leq \langle \Phi', \sigma'' \rangle$. It follows that $\langle \Phi, \theta \circ \sigma \rangle \leq \langle \Phi', \theta \circ \sigma'' \rangle$. □

Matching and unification are closely related notions. While unification is the basis of logic programming languages, matching is at the heart of rewriting and functional programming. A matching algorithm can be easily derived from a unification algorithm. First we recall the definition of matching.

**Definition 4.13.** A *matching problem* is a particular kind of unification problem $\mathtt{Pr}$ in which the variables in right-hand sides of equality constraints are disjoint from the variables in left-hand sides. A solution $\langle \Phi, \sigma \rangle$ for a matching problem $\mathtt{Pr}$ satisfies $\Phi \vdash s \stackrel{\curlywedge}{\approx}_\alpha t\sigma$ and $\Phi \vdash \pi \curlywedge t\sigma$, for each $s \stackrel{\curlywedge}{\approx}_\alpha^{?} t, \pi \curlywedge^{?} t \in \mathtt{Pr}$ (i.e., in a matching problem, only the variables in right-hand sides of terms can be instantiated.

A nominal matching algorithm can be obtained from the unification algorithm specified in Table 1 by restricting the instantiation rules, so that only variables which were in left-hand sides of equality constraints in the initial problem can be instantiated.

**Observation 4.14.** Theorem 3.12 guarantees the equivalence between $\approx_\alpha$ and $\stackrel{\curlywedge}{\approx}_\alpha$, therefore, we can associate the unification algorithm proposed, with the standard nominal unification algorithm proposed in [UPG04]. The problem $\mathtt{Pr}$ introduced in Example 4.7, is equivalent to the nominal unification problem $\mathcal{P} = \{[a]\mathtt{f}(X, a) \approx_\alpha^{?} [b]\mathtt{f}((b\ c) \cdot W, (a\ c) \cdot Y\}$, and using the standard simplification rules [UPG04]:

$$\mathcal{P} \stackrel{*}{\Longrightarrow} \stackrel{[Y \mapsto b]}{\Longrightarrow} \stackrel{*}{\Longrightarrow} \mathcal{P}' = \{X \approx_\alpha^{?} (a\ b) \cdot ((b\ c) \cdot W), a\#^{?}W\} \stackrel{[X \mapsto (a\ b) \circ (b\ c) \cdot W]}{\Longrightarrow} \{a\#^{?}W\} = \mathcal{P}' \tag{4.1}$$

The pair $\langle \mathcal{P} \rangle_{\mathtt{sol}} = \langle \{a\#W\}, \delta \rangle$, where $\delta = \{Y \mapsto b, X \mapsto (a\ b) \circ (b\ c) \cdot W\}$ 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 $\mathtt{Pr} = \{[a]\mathtt{f}(X, a) \stackrel{\curlywedge}{\approx}_\alpha^{?} [b]\mathtt{f}((b\ c) \cdot W, (a\ c) \cdot Y\}$. Similarly, from the solution $\langle \Psi, \sigma \rangle$ proposed in Example 4.10, 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. $\stackrel{\curlywedge}{\approx}_\alpha$ and $\curlywedge$, and $\mathcal{P}_\#$ denotes a unification problem w.r.t. $\approx_\alpha$ and $\#$.

**Theorem 4.15.** *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) $\langle [\Delta]_\curlywedge, \delta \rangle \in \mathcal{U}(\mathtt{Pr}_\curlywedge)$.

*Proof.* Consequence of Theorems 3.15 and 3.16 and the fact that substitution preserves #, $\approx_\alpha$, $\curlywedge$ and $\overset{\curlywedge}{\approx}_\alpha$ judgements (see Proposition 3.13). □

It is easy to see that the algorithm in Table 1 is exponential (as the one given in [UPG04]): even if we restrict the problems to first-order unification problems (without atoms), the simplification of the following problem requires a number of steps which is exponential with respect to the size of the problem.

$$h(f(X_0, X_0), ..., f(X_{n-1}, X_{n-1})) \overset{\curlywedge ?}{\approx}_\alpha h(X_1, ..., X_n)$$

Comparing this algorithm with the one based on freshness constraints, one notices that there is a correspondence between the simplification rules in both approaches (the term syntax is the same in both cases). Moreover, the simplification rules for fixed-point constraints work exactly like the simplification rules for freshness constraints (fixed-point rules have linear complexity, same as the simplification rules for freshness constraints in the freshness-based approach). The techniques designed to improve the efficiency of the freshness-based nominal unification algorithm (see, e.g., [Cal10, LV10]) can equally apply to the fixed-point based algorithm. To avoid the exponential complexity, terms should be represented via graphs, and permutations should be applied in a lazy way (see [Cal10, LV10] for details). We leave the development of efficient implementations to future work and in the rest of the paper we focus on unification modulo equational theories.

## 5. Nominal alpha-equivalence modulo equational theories

In this section an extension of $\alpha$-equality to take into account equational theories will be proposed. It is well-known that first-order unification modulo an arbitrary equational theory $\mathsf{E}$ is undecidable, therefore, it is expected that nominal unification modulo $\mathsf{E}$ (nominal $\mathsf{E}$-unification, for short) inherits this undecidability property. In this work, we propose an approach to deal with particular classes of equational theories, such as, associativity ($\mathsf{A}$), commutativity ($\mathsf{C}$) and associativity-commutativity ($\mathsf{AC}$), using fixed-point constraints.

In the case in which $\mathsf{E} = \mathsf{A}$, $\mathsf{C}$, $\mathsf{AC}$, or a combination of these theories, an algorithm to check $\mathsf{E}$-$\alpha$-equality via freshness constraints was proposed in [AdCSFN17, AdCSMFR19], where correctness results were formally verified using the Coq proof assistant, and an implementation in Ocaml was given. Unification was considered only for $\mathsf{C}$ theories, for which it was shown that in general there is no finitary representation of the set of solutions if solutions are represented by freshness contexts and substitutions.

We argue that the approach of fixed-point constraints is convenient when dealing with unification problems in equational theories that involve some notion of permutation of elements (such as commutativity), as it was shown in the previous version of this paper [AFN18].

### 5.1. Alpha-equivalence modulo $\mathsf{E}$ via permutation fixed points.

In this section the relations $\overset{\curlywedge}{\approx}_\alpha$ and $\curlywedge$ will be extended to $\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$ and $\curlywedge_\mathsf{E}$, where $\mathsf{E}$ is some equational theory. Inference rules will be parameterised by $\mathsf{E}$ so they can be reused when other theories are exploited.

In this work we have dedicated rules for $\mathsf{A}$, $\mathsf{C}$ and $\mathsf{AC}$, and their application depends on whether the signature $\Sigma$ has function symbols satisfying these theories. Whenever we want to restrict the results to a particular theory, we will explicitly indicate the $\mathsf{E}$.

Similarly to the case of pure/syntactic $\alpha$-equality we define notions of $\mathsf{E}$-fixed-point and $\mathsf{E}$-$\alpha$-equality constraints, as well as $\mathsf{E}$-fixed-point contexts and $\mathsf{E}$-judgments as expected.

**Definition 5.1** ($\mathsf{E}$-constraints and $\mathsf{E}$-fixed-point contexts)**.** • An $\mathsf{E}$-*fixed-point constraint* is a pair of the form $\pi \curlywedge_{\mathsf{E}} t$, of a permutation $\pi$ and a term $t$. An $\mathsf{E}$-$\alpha$-*equality constraint* (for short, $\mathsf{E}$-equality constraint or just equality constraint) is a pair of the form $s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$, for nominal terms $s$ and $t$.

• We call a fixed-point constraint of the form $\pi \curlywedge_{\mathsf{E}} X$ a *primitive* $\mathsf{E}$-*fixed-point constraint* and a finite set of such constraints is called an $\mathsf{E}$-*fixed-point context*. $\Upsilon, \Psi, \ldots$ range over contexts.

Intuitively, $s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$ will mean that $s$ and $t$ are $\alpha$-equivalent modulo the equational theory $\mathsf{E}$, and $\pi \curlywedge_{\mathsf{E}} t$ will mean that the permutation $\pi$ has no effect on the equivalence class of the term $t$ modulo $\mathsf{E}$. For instance, $(a\ c) \curlywedge_{\mathsf{C}} +(a, c)$, assuming $+$ is commutative, but not $(a\ c) \curlywedge_{\mathsf{C}} \mathsf{f}(a, c)$, if $\mathsf{f}$ is not a commutative symbol.

Below we assume that commutative symbols are always applied to pairs (although the grammar of nominal terms permits application of function symbols to tuples, we assume a syntactic check is carried out in the case of $\mathsf{C}$-symbols).

**Definition 5.2.** An $\mathsf{E}$-*fixed-point judgement* is a tuple $\Upsilon \vdash И\bar{c}.\pi \curlywedge_{\mathsf{E}} t$ of a fixed-point context and a fixed-point constraint, possibly with some newly quantified atoms, whereas an $\mathsf{E}$-$\alpha$-*equality judgement* is a tuple $\Upsilon \vdash И\bar{c}.s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$ of a fixed-point context and an $\mathsf{E}$-equality constraint, also with some newly quantified atoms $\bar{c}$.

$\mathsf{E}$-judgements are derived using the rules in Figures 8 and 9.

Notice that unlike the syntactical case, where the deduction rules for $\curlywedge$ do not use $\stackrel{\curlywedge}{\approx}_{\alpha}$, here the rules for $\curlywedge_{\mathsf{E}}$ and $\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$ are mutually recursive when the theory $\mathsf{E}$ involves function symbols with commutative and associative-commutativity properties (see rules ($\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}}$) and ($\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{AC}}$). We assume that the terms are flattened w.r.t. associative and associative-commutative function symbols.

Despite the fact that the rules are mutually recursive, the relations are well-defined since the recursion is well-founded. To see this, one can use a measure that interprets each equality judgement by the pair consisting of the maximum of the sizes of terms in the equality constraint and the symbol $\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$, and each fixed-point judgement by the pair consisting of the size of the term in the fixed-point constraint and the fixed-point symbol. For example, $\Upsilon \vdash И c.s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$ is interpreted by $\langle max(|s|, |t|), \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\rangle$ and $\Upsilon \vdash И c.(a\ b) \curlywedge_{\mathsf{E}} t$ by $\langle |t|, \curlywedge_{\mathsf{E}}\rangle$. We compare these pairs lexicographically, using the ordering on natural numbers to compare the sizes of terms, and the ordering $\curlywedge_{\mathsf{E}} > \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$ for the second component of pairs. We can then check that in all the rules in Figures 8 and 9 the premises are strictly smaller than the conclusion according to this ordering, therefore the rules provide an inductive definition of the set of derivable judgements.

Rules ($\curlywedge_{\mathsf{E}}\mathbf{a}$), ($\curlywedge_{\mathsf{E}}\mathbf{var}$), ($\curlywedge_{\mathsf{E}}\mathbf{abs}$) and ($\curlywedge_{\mathsf{E}}\mathbf{tuple}$) behave exactly as the corresponding rules in Figure 3 (where $\mathsf{E} = \emptyset$), i.e., the theory $\mathsf{E}$ has no effect on the fixed-point constraint.

Rule ($\curlywedge_{\mathsf{E}}\mathsf{f}$) is used for associative and uninterpreted function symbols. In the case of commutative or associative-commutative function symbols the rules ($\curlywedge_{\mathsf{E}}\mathbf{f}^{\mathsf{C}}$) and ($\curlywedge_{\mathsf{E}}\mathbf{f}^{\mathsf{AC}}$) are used. The goal is to ensure the analogous of Theorem 3.12 for the relations $\curlywedge_{\mathsf{E}}$ and $\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$

that we wish to obtain: $\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} t$ iff $\Upsilon \vdash \overline{c}.\pi \cdot t \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$ (Theorem 5.6). This is illustrated in the example below.

$$\frac{\pi(a) = a}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} a}\,(\curlywedge_{\mathsf{E}}\mathbf{a}) \qquad \frac{\mathtt{supp}(\pi^{\pi'^{-1}}) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} \pi' \cdot X}\,(\curlywedge_{\mathsf{E}}\mathbf{var})$$

$$\frac{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} t}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} \mathsf{f}^{\mathsf{E}} t}\,(\curlywedge_{\mathsf{E}}\mathsf{f}), \text{if } \mathsf{f}^{\mathsf{E}} = \mathsf{f}, \mathsf{f}^{\mathsf{A}} \qquad \frac{\Upsilon \vdash И\overline{c}, c_1.\pi \curlywedge_{\mathsf{E}} (a\ c_1) \cdot t}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} [a]t}\,(\curlywedge_{\mathsf{E}}\mathbf{abs})$$

$$\frac{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} t_1 \quad \ldots \quad \Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{C}} t_n}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} (t_1, \ldots, t_n)}\,(\curlywedge_{\mathsf{E}}\mathbf{tuple})$$

$$\frac{\Upsilon \vdash И\overline{c}.\pi \cdot (\mathsf{f}^{\mathsf{C}}(t_0, t_1)) \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)}\,(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$$

$$\frac{\Upsilon \vdash И\overline{c}.\pi \cdot (\mathsf{f}^{\mathsf{AC}}(t_0, t_1, \ldots, t_n)) \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(t_0, t_1, \ldots, t_n)}{\Upsilon \vdash И\overline{c}.\pi \curlywedge_{\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(t_0, t_1, \ldots, t_n)}\,(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{AC}})$$

Figure 8: Fixed-point rules modulo A, C, AC.

$$\frac{}{\Upsilon \vdash И\overline{c}.a \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} a}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathbf{a}) \qquad \frac{\mathtt{supp}((\pi')^{-1} \circ \pi) \setminus \{\overline{c}\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash И\overline{c}.\pi \cdot X \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \pi' \cdot X}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathbf{var})$$

$$\frac{\Upsilon \vdash И\overline{c}.s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} (a\ b) \cdot t \quad \Upsilon \vdash И\overline{c}, c_1.(a\ c_1) \curlywedge_{\mathsf{E}} t}{\Upsilon \vdash И\overline{c}.[a]s \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} [b]t}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathbf{ab}) \qquad \frac{\Upsilon \vdash И\overline{c}.t \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t'}{\Upsilon \vdash И\overline{c}.[a]t \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} [a]t'}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}[\mathbf{a}])$$

$$\frac{\Upsilon \vdash И\overline{c}.t \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t'}{\Upsilon \vdash И\overline{c}.\mathsf{f}^{\mathsf{E}} t \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{E}} t'}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathsf{f}), \begin{array}{l}\text{if } \mathsf{f}^{\mathsf{E}} = \mathsf{f}, \mathsf{f}^{\mathsf{A}} \text{ or}\\ t, t' \text{ are not pairs}\end{array}$$

$$\frac{\Upsilon \vdash И\overline{c}.t_1 \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t'_1 \quad \ldots \quad \Upsilon \vdash И\overline{c}.t_n \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t'_n}{\Upsilon \vdash И\overline{c}.(\widetilde{t})_{1..n} \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} (\widetilde{t'})_{1..n}}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathbf{t})$$

$$\frac{\Upsilon \vdash И\overline{c}.s_0 \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_i \quad \Upsilon \vdash И\overline{c}.s_1 \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_{(i+1)\ \bmod 2}}{\Upsilon \vdash И\overline{c}.\mathsf{f}^{\mathsf{C}}(s_0, s_1) \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathsf{f}^{\mathsf{C}}) \text{ where } i \in \{0,1\}$$

$$\frac{\Upsilon \vdash И\overline{c}.s_0 \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_i \quad \Upsilon \vdash И\overline{c}.\mathsf{f}^{\mathsf{AC}}(\widetilde{s})_{1..n} \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(\widetilde{t})^{-i}_{0..n}}{\Upsilon \vdash И\overline{c}.\mathsf{f}^{\mathsf{AC}}(\widetilde{s})_{0..n} \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(\widetilde{t})_{0..n}}\,(\stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\mathsf{f}^{\mathsf{AC}}) \begin{array}{l}\text{where } \ i \in \{0,\ldots,n\},\\ (\widetilde{t_0})^{-i}_n = (t_0, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)\end{array}$$

Figure 9: Rules for equality modulo A, C, AC

**Example 5.3.** Let $+$ be a commutative function symbol and suppose that we want to decide whether $\emptyset \vdash (a\ b) \curlywedge_{\mathsf{C}} ((a+b)+c)$, which corresponds to $\emptyset \vdash (a\ b) \cdot ((a+b)+c) \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{C}} (a+b)+c$ since $(a\ b) \cdot ((a+b)+c) = (b+a)+c \stackrel{\curlywedge}{\approx}_{\alpha,\mathsf{C}} (a+b)+c$. In general, $\Upsilon \vdash \pi \curlywedge_{\mathsf{C}} t_0 + t_1$

means that the permutation $\pi$ fixes $t_0 + t_1$ modulo $\mathsf{C}$ (given the information in $\Upsilon$), that is, $\Upsilon \vdash \pi \cdot (t_0 + t_1) \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_0 + t_1$. By definition, the permutation $\pi$ distributes homomorphically over the operator $+$, therefore, we have $\Upsilon \vdash \pi \cdot t_0 + \pi \cdot t_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_0 + t_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_1 + t_0$. Thus, two cases can be distinguished: $\Upsilon \vdash \pi \cdot t_i \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_i$ or $\Upsilon \vdash \pi \cdot t_i \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_{(i+1) mod\ 2}$, for $i = 0, 1$ (see rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}}\ \mathbf{f}^{\mathsf{C}})$ in Figure 9).

Similarly, $\alpha$-equality rules (see Figure 4) have their equational counterpart in Figure 9: rules for atoms, tuples and abstractions are not affected by the theory $\mathsf{E}$, whereas rules involving function symbols $\mathsf{f}^{\mathsf{E}}$ have to be analysed separately.

- Associative symbols: rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{A}})$ assumes that terms are flattened with respect to nested occurrences of $\mathsf{f}^{\mathsf{A}}$.
- Commutative symbols: rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{C}})$ is used.
- Associative-commutative symbols: terms are assumed to be flattened with respect to the $\mathsf{f}^{\mathsf{AC}}$ function symbol, and rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{AC}})$ is used.

**Example 5.4.** Consider the signature $\Sigma_{\mathsf{A}} = \{\mathsf{f}^{\mathsf{A}}\} \cup \Sigma^{\emptyset}$, where $\Sigma^{\emptyset}$ is a set of uninterpreted function symbols. Rules $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{AC}})$, $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$, $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{C}})$ and $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{AC}})$ will not be used in judgements involving $\Sigma_{\mathsf{A}}$-terms, therefore, we can replace $\mathsf{E}$ for $\mathsf{A}$ and obtain rules for $\overset{\curlywedge}{\approx}_{\alpha,\mathsf{A}}$ and $\curlywedge_{\mathsf{A}}$. For instance, consider $\mathsf{f}^{\mathsf{A}}([a]a, \mathsf{f}^{\mathsf{A}}(b, X))$, which is represented in flattened form as $\mathsf{f}^{\mathsf{A}}([a]a, b, X)$.

$$\dfrac{\dfrac{\dfrac{}{\Upsilon \vdash \mathit{И}c_1.(a\ d)\curlywedge_{\mathsf{A}} c_1}(\curlywedge_{\mathsf{A}}\mathbf{a})}{\Upsilon \vdash (a\ d)\curlywedge_{\mathsf{A}} [a]a}(\curlywedge_{\mathsf{A}}\mathbf{abs}) \quad \dfrac{}{\Upsilon \vdash (a\ d)\curlywedge_{\mathsf{A}} b}(\curlywedge_{\mathsf{A}}\mathbf{a}) \quad \dfrac{\{a,d\} \subseteq \mathtt{supp}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash (a\ d)\curlywedge_{\mathsf{A}} X}(\curlywedge_{\mathsf{A}}\mathbf{a})\ \ ?}{\Upsilon \vdash (a\ d)\curlywedge_{\mathsf{A}} ([a]a, b, X)}(\curlywedge_{\mathsf{A}}\mathsf{tuple})}{\Upsilon \vdash (a\ d)\curlywedge_{\mathsf{A}} \mathsf{f}^{\mathsf{A}}([a]a, b, X)}(\curlywedge_{\mathsf{A}}\mathsf{f}^{\mathsf{A}})$$

The conclusion of this derivation depends on the support of the permutations in $\Upsilon$, this is illustrated with the question mark '**?**' in the rightmost leaf in the derivation. For example, if, on one hand, $\Upsilon = \emptyset$, then this derivation fails and we cannot conclude that $(a\ d)$ fixes $\mathsf{f}^{\mathsf{A}}([a]a, b, X)$; if, on the other hand, $\Upsilon = \{(a\ b) \curlywedge X, (d\ e) \curlywedge X\} \cup \Upsilon'$, then we could conclude the opposite.

**Example 5.5.** Consider the signature $\Sigma_{\{\mathsf{C},\mathsf{AC}\}} = \{\oplus^{\mathsf{C}}, \mathsf{or}^{\mathsf{AC}}\} \cup \{\forall^{\emptyset}, \mathsf{g}^{\emptyset}\} \cup \Sigma^{\emptyset}$, where $\forall$ and $\mathsf{g}$ are unary uninterpreted function symbols. To improve readability, we will omit the superscripts of the function symbols in the rest of this example. Since we only have commutative and associative-commutative symbols, we will replace $\mathsf{E}$ in the rules by $\{\mathsf{C}, \mathsf{AC}\}$, therefore, obtaining rules for $\overset{\curlywedge}{\approx}_{\alpha,\{\mathsf{C},\mathsf{AC}\}}$ and $\curlywedge_{\{\mathsf{C},\mathsf{AC}\}}$. By applying the rules one can verify that

- $\emptyset \not\vdash (a\ b)(b\ c) \curlywedge_{\{\mathsf{C},\mathsf{AC}\}} (\mathsf{g}(a) \oplus \mathsf{g}(b)) \oplus \mathsf{g}(c)$, this is due to the fact that $\oplus$ is commutative but not associative, and the permutation $(a\ b)(b\ c)$ swaps the atom $a$ which is an argument of the inner $\oplus$ with the atom $c$ which is an argument of the outer $\oplus$;
- $\emptyset \vdash (a\ b)(b\ c) \curlywedge_{\{\mathsf{C},\mathsf{AC}\}} \mathsf{or}(\mathsf{or}(\mathsf{g}(a), \mathsf{g}(b)), \mathsf{g}(c))$;
- $\emptyset \vdash (a\ b) \curlywedge_{\{\mathsf{C},\mathsf{AC}\}} \forall[a]\mathsf{or}(\mathsf{or}((a \oplus b), (b \oplus c)), (a \oplus c))$;
- $(a\ b)\curlywedge_{\{\mathsf{C},\mathsf{AC}\}} X \vdash \forall[a]\mathsf{or}(\mathsf{or}((a \oplus X), g(c)), g(a)) \overset{\curlywedge}{\approx}_{\alpha,\{\mathsf{C},\mathsf{AC}\}} \forall[b]\mathsf{or}(\mathsf{or}(g(c), ((a\ b)\cdot X \oplus b)), g(b))$.

The theorem below extends Theorem 3.12, relating fixed-point constraints to fixed-point equalities, for the case in which equational theories $\mathsf{A}, \mathsf{C}$ and $\mathsf{AC}$ are involved.

**Theorem 5.6.** *Let $\Upsilon, \pi$ and $t$ be an $\mathsf{E}$-fixed-point context, a permutation and a nominal term, respectively. $\Upsilon \vdash И\bar{c}.\pi \curlywedge_{\mathsf{E}} t$ iff $\Upsilon \vdash И\bar{c}.\pi \cdot t \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathsf{E}} t$.*

*Proof.* The proof is by induction on the derivation, similar to the proof of Theorem 3.12, except for the use of rules dealing with function symbols modulo $\mathsf{E}$. The rule for associative symbols is the same as for syntactic symbols; we consider the cases of commutative and associative-commutative symbols.

- Suppose that $\Upsilon \vdash И\bar{c}.\pi \curlywedge_{\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)$, therefore, rule $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$ was applied, and one gets $\Upsilon \vdash И\bar{c}.\pi \cdot (\mathsf{f}^{\mathsf{C}}(t_0, t_1)) \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)$, and the result follows trivially. The other direction is also trivial.
- Suppose that $\Upsilon \vdash И\bar{c}.\pi \curlywedge_{\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(t_0, \ldots, t_n)$, therefore, rule $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{AC}})$ was applied, and one gets $\Upsilon \vdash И\bar{c}.\pi \cdot (\mathsf{f}^{\mathsf{AC}}(t_0, \ldots, t_n)) \mathrel{\overset{\curlywedge}{\approx}}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{AC}}(t_0, \ldots, t_n)$, and the result follows trivially. The other direction is also trivial. □

5.2. **From freshness to $\mathsf{E}$-fixed-point constraints and back again.** In [ARdCSFNS18] relations $\approx_{\{\alpha,\mathsf{A}\}}$, $\approx_{\{\alpha,\mathsf{C}\}}$, $\approx_{\{\alpha,\mathsf{AC}\}}$ and their combination $\approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}}$ were defined as extensions of $\approx_\alpha$ using the standard approach via freshness constraints (see the rules in Figures 1 and 2) but using specific rules for associative, commutative and associative-commutative symbols. We recall those rules in Figure 10.

$$\frac{\nabla \vdash s \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} t}{\nabla \vdash \mathsf{f}^{\mathsf{E}}\, s \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{\mathsf{E}}\, t}\ (\approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \textbf{app}),\ \text{if } \mathsf{E} \notin \{\mathsf{A}, \mathsf{C}, \mathsf{AC}\} \text{ or both } s \text{ and } t \text{ are not pairs}$$

$$\frac{\nabla \vdash s_0 \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} t_0 \qquad \mathsf{f}^{\mathsf{A}}(\widetilde{s})_{1..n} \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{\mathsf{A}}(\widetilde{t})_{1..n}}{\nabla \vdash \mathsf{f}^{\mathsf{A}}(\widetilde{s})_{0..n} \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{\mathsf{A}}(\widetilde{t})_{0..n}}\ (\approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{A})$$

$$\frac{\nabla \vdash s_0 \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} t_i \quad \nabla \vdash s_1 \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} t_{(i+1)\,mod\,2} \quad i = 0, 1}{\nabla \vdash \mathsf{f}^{\mathsf{C}}(s_0, s_1) \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)}\ (\approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{C})$$

$$\frac{\nabla \vdash s_0 \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} t_i \quad \nabla \vdash \mathsf{f}^{\mathsf{AC}}(\widetilde{s})_{1..n} \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{AC}(\widetilde{t})^{-i}_{0..n}}{\nabla \vdash \mathsf{f}^{\mathsf{AC}}(\widetilde{s})_{0..n} \approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{f}^{\mathsf{AC}}(\widetilde{t})_{0..n}}\ (\approx_{\{\mathsf{A},\mathsf{C},\mathsf{AC}\}} \mathsf{AC})$$

Figure 10: Additional rules for equational $\alpha$-equivalence via freshness constraints

Using a generalisation of the functions $[\_]_{\curlywedge}$ and $[\_]_{\#}$ defined in Section 3.3, we can obtain results that extend Theorem 3.15 and Theorem 3.16 to the equational case. The functions $[\_]^{\mathsf{E}}_{\curlywedge}$ and $[\_]^{\mathsf{E}}_{\#}$ defined below are the natural extension of the previous translation functions.

The mapping $[\_]^{\mathsf{E}}_{\curlywedge}$ associates each primitive freshness constraint in $\Delta$ with a primitive fixed-point constraint:

$$[a\#X]^{\mathsf{E}}_{\curlywedge} = (a\ c_a) \curlywedge_E X \text{ where } c_a \text{ is a new name}$$

This mapping extends directly to contexts. We denote by $[\Delta]^{\mathsf{E}}_{\curlywedge}$ the image of $\Delta$ under $[\_]^{\mathsf{E}}_{\curlywedge}$.

The mapping $[\_]^{\mathsf{E}}_{\#}$ associates each primitive fixed-point constraint in $\Upsilon$ with a primitive freshness context:

$$[\pi \curlywedge_{\mathsf{E}} X]^{\mathsf{E}}_{\#} = \mathtt{supp}(\pi)\#X.$$

We denote by $[\Upsilon]^{\mathsf{E}}_{\#}$ the union of the freshness contexts obtained by translating each constraint in $\Upsilon$ using $[\_]^{\mathsf{E}}_{\#}$.

**Theorem 5.7.** (1) $\Delta \vdash a\#t \Leftrightarrow [\Delta]^{\mathsf{E}} \vdash И c.(a\ c) \curlywedge_{\mathsf{E}} t$.
(2) $\Upsilon \vdash И\bar{c}.\pi \curlywedge_{\mathsf{E}} t \Leftrightarrow [\Upsilon]^{\mathsf{E}}_{\#} \vdash \overline{\mathit{supp}(\pi) \setminus \{\bar{c}\}}\#t$.

*Proof.* The proof follows the same lines of the proof of Theorem 3.15. We discuss only the proof of part (1). The proof is by induction on rules of Figure 1 used in the derivation of $\Delta \vdash a\#t$. We show only the cases corresponding to function symbols.
($\Longrightarrow$)

- Rule ($\#\mathbf{f}$)
  In this case $t = \mathsf{f}^{\mathsf{E}}t'$, for some theory $\mathsf{E}$. The analysis is based on the specific theory:
  (1) $\mathsf{f}^{\mathsf{E}} = \mathsf{f}^{\mathsf{A}}$
  This case is analogous to the case in which $\mathsf{E} = \emptyset$.
  (2) $\mathsf{f}^{\mathsf{E}} = \mathsf{f}^{\mathsf{C}}$
  There is a proof $\Pi$ of the form
  $$\dfrac{\dfrac{\Pi}{\Delta \vdash a\#(t_1,t_2)}}{\Delta \vdash a\#\mathsf{f}^{\mathsf{C}}(t_1,t_2)}\,(\#\mathbf{f})$$
  By induction hypothesis, there exists a proof $\Pi'$ of $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \curlywedge_{\mathsf{E}} (t_1,t_2)$. Therefore, there is a proof of the form
  $$\dfrac{\dfrac{\Pi_1'}{[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \curlywedge_{\mathsf{E}} t_1} \qquad \dfrac{\Pi_2'}{[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \curlywedge_{\mathsf{E}} t_2}}{[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \curlywedge_{\mathsf{E}} (t_1,t_2)}\,(\curlywedge_{\mathsf{E}}\mathbf{tuple})$$
  From $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \curlywedge_{\mathsf{E}} t_i$ one can derive that $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \cdot t_i \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_i$, by Theorem 5.6, for $(i = 1,2)$. By applying rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}\ \mathsf{f}^{\mathsf{C}})$, it follows that
  $$[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c_a.(a\ c_a) \cdot (\mathsf{f}^{\mathsf{C}}(t_1,t_2)) \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{C}}(t_1,t_2),$$
  and the result follows from Theorem 5.6.
  (3) $\mathsf{f}^{\mathsf{E}} = \mathsf{f}^{\mathsf{A}}$
  The proof is analogous to the case above.

($\Longleftarrow$) The interesting case is again for rule $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$.

Suppose that $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \curlywedge_{\mathsf{E}} t_1 \oplus t_2$. We want to prove that $\Delta \vdash a\#t_1 \oplus t_2$. From rule $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$ we can conclude that

- either there exist proofs of $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \cdot t_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_1$ and $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \cdot t_2 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_2$, and by Theorem 5.6 it follows that there exist proofs of $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \curlywedge_{\mathsf{E}} t_1$ and $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \curlywedge_{\mathsf{E}} t_2$, and the result follows by induction hypothesis.
- or there exist proofs of $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \cdot t_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_2$ and $[\Delta]^{\mathsf{E}}_{\curlywedge} \vdash И c.(a\ c) \cdot t_2 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t_1$.
  Since these equalities hold for any new name $c$ (not occurring in $t_1$ or $t_2$), then $a$ cannot be in the support of $t_1$ and $t_2$ and therefore $a$ is fresh in $t_1$ and $t_2$. □

We can now relate $\alpha$-equivalence modulo $\mathsf{E}$ via freshness constrains ($\approx_{\{\alpha,\mathsf{E}\}}$) with its version via fixed-point constraints ($\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}}$).

**Theorem 5.8.** (1) $\Upsilon \vdash \mathit{\forall}\overline{c}.\ s \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t \Rightarrow [\Upsilon]^{\mathsf{E}}_{\#} \cup \Delta \vdash s \approx_{\{\alpha,\mathsf{E}\}} t$, *where* $\Delta \vdash \overline{c} \# \mathit{Var}(s,t)$.
(2) $\Delta \vdash s \approx_{\{\alpha,\mathsf{E}\}} t \Rightarrow [\Delta]^{\mathsf{E}}_{\curlywedge} \vdash s \overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} t$.

*Proof.* The proof is very similar to the proof of Theorem 3.16, except for the case of rules involving function symbols $\mathsf{f}^{\mathsf{E}}$, with $\mathsf{E} \neq \emptyset$, where the reasoning is similar to the one in the proof of Theorem 5.7. □

5.3. **Solving nominal C-unification problems 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

$$+((a\ b)\cdot X, a) \overset{\curlywedge ?}{\approx}_{\alpha} +(Y, X) \tag{5.1}$$

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 [ARdCSFNS18], a simplification algorithm for solving nominal C-unification was proposed. This algorithm was based on the standard nominal unification algorithm [UPG04] where $\alpha$-equivalence is defined w.r.t. the notion of freshness. Upon the input of a unification problem $\mathtt{Pr}$, the algorithm outputs a finite family of triples of the form $\langle \nabla, \sigma, P\rangle$, where $\nabla$ is a freshness context, $\sigma$ a substitution and $P$ is a set of fixed-point equations, which are solved using a separate procedure.

In [ARdCSFNS17] it is proved that even a simple unification problem such as $(a\ b)\cdot X \approx_{\alpha} X$ (i.e., a problem consisting of just one fixed-point equation) could produce an infinite and independent set of solutions, whenever the signature contains commutative function symbols. For example, if $f$ is commutative, the following substitutions solve this equation: $\{X \mapsto a+b, X \mapsto f(a+b), X \mapsto [e](a+b, b+a), \ldots\}$. Therefore, it is not possible to obtain a finite and complete set of solutions for every solvable unification problem if solutions are expressed using freshness constraints and substitutions. However, we remark that the problem $+((a\ b)\cdot X, a) \overset{\curlywedge ?}{\approx}_{\alpha} (Y, X)$ 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 $\{Y \mapsto a, (a\ b) \curlywedge X\}$. This observation led us to use fixed-point constraints instead of freshness constraints to express solutions.

Similarly to Section 4, below we define the notion of nominal C-unification in terms of C-fixed-point constraints, and provide a nominal C-unification unification algorithm specified by means of simplification rules.

In this section, as in Section 4, we assume a generator of new names exists, and remove the new quantifier from the syntax of unification problems.

**Definition 5.9.** A C-*unification problem* $\mathtt{Pr}$ is a pair $\langle \Phi, P\rangle$ where $P$ is a finite set of C-equality constraints $s \overset{\curlywedge ?}{\approx}_{\mathsf{C}} t$ and $\Phi$ is a finite set of C-fixed-point constraints $\pi \curlywedge^{?}_{\mathsf{C}} t$. To ease the notation, we will denote $s \overset{\curlywedge ?}{\approx}_{\mathsf{C}} t$ by $s \approx^{?} t$.

**Definition 5.10** (Solutions of C-unification problems)**.** A solution for a C-unification problem $\mathtt{Pr} = \langle \Phi, P\rangle$ is a pair $\langle \Upsilon, \sigma\rangle$, where the following conditions are satisfied

(1) $\Upsilon \vdash \pi \curlywedge_{\mathsf{C}} t\sigma$, if $\pi \curlywedge^{?}_{\mathsf{C}} t \in \Phi$;
(2) $\Upsilon \vdash s\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t\sigma$, if $s \approx^{?} t \in P$.
(3) $\Upsilon \vdash X\sigma\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} X\sigma$.

The set of solutions for a $\mathsf{C}$-unification problem $\mathtt{Pr}$ is denoted as $\mathcal{U}_{\mathsf{C}}(\mathtt{Pr})$.

**Definition 5.11** (Most general solution and complete set of solutions)**.**
- For $\langle \Upsilon, \sigma\rangle$ and $\langle \Psi, \delta\rangle$ in $\mathcal{U}_{\mathsf{C}}(\mathtt{Pr})$, we say that $\langle \Upsilon, \sigma\rangle$ is *more general than* $\langle \Psi, \delta\rangle$, denoted $\langle \Upsilon, \sigma\rangle \preceq \langle \Psi, \delta\rangle$, if there exists a substitution $\rho$ satisfying $\Psi \vdash \sigma\rho \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} \delta$ and $\Psi \vdash \Upsilon\rho$.
- A subset $\mathcal{C}$ of $\mathcal{U}_{\mathsf{C}}(\mathtt{Pr})$ is a *complete set of solutions* of $\mathtt{Pr}$ if for all $\langle \Psi, \sigma\rangle \in \mathcal{U}_{\mathsf{C}}(\mathtt{Pr})$, there exits a $\langle \Upsilon, \delta\rangle \in \mathcal{C}$ such that $\langle \Upsilon, \delta\rangle \preceq \langle \Psi, \sigma\rangle$. We denote a complete set of solutions of the $\mathsf{C}$-unification problem $\mathtt{Pr}$ as $\mathcal{C}(\mathtt{Pr})$.

Table 2 presents the simplification rules for $\mathsf{C}$-unification problems. They are derived from the deduction rules for judgements, as done for the syntactic case. The main difference is that now there are two rules for the simplification of fixed-point constraints involving commutative symbols (rules $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}\mathbf{1})$ and $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}\mathbf{2})$) and two rules to deal with equality of terms rooted by a commutative symbol (rules $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}}\ \mathsf{f}^{\mathsf{C}}\mathbf{1})$ and $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}}\ \mathsf{f}^{\mathsf{C}}\mathbf{2})$).

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

Table 2: Simplification Rules for $\mathsf{C}$-unification problems via $\mathsf{C}$-fixed-point constraints. In rules $(\curlywedge_{C} abs)$ and $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}}\ abs2)$, $c_1$ and $c_2$ are newly generated names.

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

**Lemma 5.12** (Termination of simplification for C-unification problems)**.** *There is no infinite chain of reductions* $\Longrightarrow_{\mathsf{C}}$ *starting from a* C*-unification 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 *heights* 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 primitive ones and/or constraints where terms have smaller height. □

The simplification rules (Table 2) specify a C-*unification algorithm*: we apply the simplification rules in a problem $\mathtt{Pr}$ until we reach normal forms. In the case of a term rooted by a commutative symbol, two rules can be applied, so a tree of derivations is built. The termination property (Lemma 5.12) guarantees the tree is finite.

For the leaves in the tree (i.e., normal forms), the notions of *consistency, failure, correctness* can be defined as in Section 4 (see Definition 4.8). So, if a normal form contains equality constraints, or inconsistent fixed-point constraints of the form $\pi \curlywedge^{?}_{\mathsf{C}} a$ such that $\pi(a) \neq a$ then this normal form is a failure. Only leaves containing consistent fixed-point constraints produce solutions.

We now prove that the C-unification algorithm is sound and complete. The proof is done in two stages, first we show that the non-instantiating rules preserve solutions if we consider all the branches of the derivation tree (here it is important to consider all the branches: due to the non-deterministic application of rules involving commutative operators, if we consider just one branch we may loose solutions). Then we show that the set of solutions computed from all the successful leaves is a complete set of solutions for the initial problem.

**Lemma 5.13** (Correctness of non-instantiating rules)**.** *Let* $\mathtt{Pr}$ *be a* C*-unification problem and* $n$ *a natural number. Assume* $\mathtt{Pr} \overset{n}{\Longrightarrow}_{\mathsf{C}} \mathtt{Pr}'_i$ $(i \in I)$ *are all the reduction sequences of length smaller than or equal to* $n$ *starting from* $\mathtt{Pr}$ *that do not use instantiating rules* $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} inst1)$ *and* $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} inst2)$*. Then*
(1) $\mathcal{U}_{\mathsf{C}}(\mathtt{Pr}) = \bigcup_{i\in I} \mathcal{U}_{\mathsf{C}}(\mathtt{Pr}'_i)$*, and*
(2) *if* $\mathtt{Pr}'_i$ *contains inconsistent reduced fixed-point constraints then* $\mathcal{U}_{\mathsf{C}}(\mathtt{Pr}'_i) = \emptyset$*.*

*Proof.* For part (1), as for the proof of Lemma 4.11, we proceed by induction on $n$, but here we need to consider all the branches of length smaller than or equal to $n$ in the derivation tree to ensure completeness.

The interesting cases are for the rules involving C function symbols, all the other cases are very similar to the proof of Lemma 4.11.

- Suppose that the last step of a simplification chain has the form

$$\mathtt{Pr}_{n-1} = \mathtt{Pr}' \uplus \{\pi \curlywedge^{?}_{\mathsf{C}} \mathsf{f}^{\mathsf{C}}(s_0, s_1)\} \Longrightarrow \mathtt{Pr}' \cup \{\pi \cdot s_0 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}}{}^{?} s_i, \pi \cdot s_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} s_{(i+1) mod 2}\} = \mathtt{Pr}^i_n.$$

  In this case, the rule used is either $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}1)$ or $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}2)$.

  Assume $i = 0$ and $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}1)$ was used (the case $i = 1$ is identical).

  There is another reduction sequence of the same length using $(\curlywedge_{\mathsf{C}}\mathsf{f}^{\mathsf{C}}2)$ and ending on $\mathtt{Pr}^1_n$ (the same problem with $i = 1$).

  Let $\langle \Psi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$ be a solution for $\mathtt{Pr}_{n-1}$:

(1) $\Psi \vdash \pi' \curlywedge_{\mathsf{C}} t\sigma$, for all $\pi' \curlywedge^{?} t \in \mathtt{Pr}'$ and $\Psi \vdash \pi \curlywedge_{\mathsf{C}} \mathsf{f}^{\mathsf{C}}(s_0, s_1)\sigma$.
(2) $\Psi \vdash t\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} s\sigma$, for all $t \approx^{?} s \in \mathtt{Pr}'$.

Since $\Psi \vdash \pi \curlywedge_{\mathsf{C}} \mathsf{f}^{\mathsf{C}}(s_0, s_1)\sigma$ and $\mathsf{f}^{\mathsf{C}}(s_0, s_1)\sigma = \mathsf{f}^{\mathsf{C}}(s_0\sigma, s_1\sigma)$, it follows that $\Psi \vdash \pi \curlywedge_{\mathsf{C}} \mathsf{f}^{\mathsf{C}}(s_0\sigma, s_1\sigma)$. From $(\curlywedge_{\mathsf{E}}\mathsf{f}^{\mathsf{C}})$, one has that either there exist a proof for $\Psi \vdash \pi \cdot s_0\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} s_i\sigma$ and $\Psi \vdash \pi \cdot s_1\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} s_{(i+1)mod\ 2}\sigma$, for $i = 0$ or for $i = 1$. Therefore, $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_n^0)$ or $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_n^1)$ and the result follows.

The other direction is similar: if $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_n^i)$, then $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$.

- Suppose the last step of the simplification chain has the form

$$\mathtt{Pr}_{n-1} = \mathtt{Pr}' \uplus \{\mathsf{f}^{\mathsf{C}}(s_0, s_1) \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} \mathsf{f}^{\mathsf{C}}(t_0, t_1)\} \Longrightarrow \mathtt{Pr}_n^i$$

where $\mathtt{Pr}_n^i = \mathtt{Pr}' \cup \{s_0 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_i, s_1 \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_{(i+1)mod\ 2}\}$.

As above, it follows that there two branches of this form, for $i = 0$ and $i = 1$.

Let $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$ be a solution for $\mathtt{Pr}_{n-1}$. In particular, $\Psi \vdash \mathsf{f}^{\mathsf{C}}(s_0\sigma, s_1\sigma) \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} \mathsf{f}^{\mathsf{C}}(t_0\sigma, t_1\sigma)$. By applying rule $(\overset{\curlywedge}{\approx}_{\alpha,\mathsf{E}} \mathsf{f}^{\mathsf{C}})$, one has that there exist proofs of $\Psi \vdash s_0\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_i\sigma$ and $\Psi \vdash s_1\sigma \overset{\curlywedge}{\approx}_{\alpha,\mathsf{C}} t_{(i+1)mod\ 2}\sigma$, for $i = 0$ or $i = 1$. Therefore, $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_n^i)$ for $i = 0$ or $i = 1$, and the result follows.

Similarly, if $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_n^i)$, then $\langle \Psi, \sigma\rangle \in \mathcal{U}(\mathtt{Pr}_{n-1})$.

The second part of the lemma follows directly from the fact that inconsistent constraints are not derivable, therefore have no solutions. □

As in Section 4, when $\mathtt{Pr}$ is a successful leaf, $\langle \mathtt{Pr}\rangle_{sol}$ consists of the composition $\sigma$ of all substitutions applied through the simplification steps and the fixed point context obtained.

**Theorem 5.14** (Soundness and Completeness)**.** *Let $\mathit{Pr} = \langle \Upsilon, P\rangle$ be a $\mathsf{C}$-unification problem and let $\{\mathit{Pr}'_i \mid \mathit{Pr} \overset{*}{\Longrightarrow}_{\mathsf{C}} \mathit{Pr}'_i$ and $\mathit{Pr}'_i$ successful normal form$\}$ be the set of all the successful normal forms of $\mathit{Pr}$ (i.e., leaves in the derivation tree without equality constraints or inconsistent fixed-point constraints).*

(1) *If $\langle \Phi, \sigma\rangle \in \bigcup\langle \mathit{Pr}'_i\rangle_{sol}$ then $\langle \Phi, \sigma\rangle \in \mathcal{U}(\mathit{Pr})$, and*
(2) *If $\langle \Phi, \sigma\rangle \in \mathcal{U}(\mathit{Pr})$, there exists $\langle \Phi', \sigma'\rangle$ such that $\langle \Phi', \sigma'\rangle \in \bigcup\langle \mathit{Pr}'_i\rangle_{sol}$ and $\langle \Phi', \sigma'\rangle \preceq \langle \Phi, \sigma\rangle$, that is, the set $\bigcup\langle \mathit{Pr}'_i\rangle_{sol}$ is a complete set of solutions.*

*Proof.* The proof is by induction on the length of a derivation $\mathtt{Pr} \Longrightarrow^{*} \mathtt{Pr}'_i$, distinguishing cases according to the first rule used. By Lemma 5.13, it is sufficient to check generality of solutions after each application of instantiation rules. If the first step uses a non-instantiating rule, then the previous lemma, together with the induction hypothesis, ensures that the set of solutions of $\mathtt{Pr}$ is exactly the set of solutions of its children. If the first step is instantiating, we proceed as in the proof of Theorem 4.12. □

**Observation 5.15.** Using the approach to nominal $\alpha$-equivalence via freshness, a nominal $\mathsf{C}$-unification algorithm was presented in [ARdCSFNS18, ARdCSFNS17], which outputs solutions represented as triples $\langle \nabla, \sigma, P\rangle$ consisting of a freshness context $\nabla$, a substitution $\sigma$ and a set $P$ of fixed-point equations of the form $\pi \cdot X \approx^{?}_{\{\alpha,\mathsf{C}\}} X$.

As with standard nominal unification, one can use the functions $[\_]_{\#}$ and $[\_]_{\curlywedge}$ to translate solutions $\langle \nabla, \sigma, P\rangle$ of nominal $\mathsf{C}$-unification problems with freshness constraints as solutions $\langle [\nabla]_{\curlywedge} \cup \{P_{\curlywedge_{\mathsf{C}}}\}, \sigma\rangle$ of nominal $\mathsf{C}$-unification problems via $\mathsf{C}$-fixed-point constraints, where $P_{\curlywedge_{\mathsf{C}}} = \{\pi \curlywedge_{\mathsf{C}} X \mid \pi \cdot X \approx^{?}_{\{\alpha,\mathsf{C}\}} X \in P\}$.

A set of simplification rules generalising the C-unification algorithm to take into account A and AC symbols was proposed in Ribeiro's thesis [dCS19] following the freshness constraint approach. The main difficulty is in the rules to deal with the AC symbols and with treatment of fixed-point equations of the form $\pi \cdot X \approx^{?}_{\{\alpha,\mathsf{AC}\}} X$. Analytical proofs of soundness and completeness of such rules were given, and a formalisation in Coq was developed for the C-unification algorithm presented in [ARdCSFNS18]. In future work we will consider and relate AC-nominal unification with freshness and fixed-point constraints.

Regarding the complexity of the C-unification algorithm based on fixed-point constraints, we observe that in the syntactic case (i.e., without $\alpha$-equivalence rules), the C-unification problem is NP-complete so it is expected that the algorithm will be exponential (Chapter 10 on Equational Unification in [BN98] surveys in detail the cases of C and AC unification). Comparing the nominal unification modulo C based on freshness and on fixed-point constraints, we can again notice that there is a one-to-one correspondence in the simplification rules, and thus the algorithms have the same behaviour during the simplification phase. The main difference is that using the freshness approach, a second algorithm is needed to solve fixed-point equations (generating an infinite number of solutions in general), which is avoided with the fixed-point approach.

## 6. Conclusions and Future Work

The notion of fixed-point constraint 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., [AK16, Cal13, CF08, Che10, FGM04]), and in particular to their extensions modulo commutativity, for which only infinite representations were possible in the standard approach. We have shown that with the new proposed approach the development of an algorithm to solve nominal equational problems modulo C is simpler, avoiding, thanks to 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. Future study will also address handling the case of Mal'cev permutative theories, which include $n$-ary functions with permutative arguments [Com93], as well as the more general and complex case of permutative equational theories [Sch89]. Finally, exploring the relation between Higher-Order Pattern unification in the style of Levy and Villaret [LV08] and nominal unification with fixed-point constrains would be also of great interest.

## References

[AdCSFN17] M. Ayala-Rincón, W. de Carvalho Segundo, M. Fernández, and D. Nantes-Sobrinho. A formalisation of nominal $\alpha$-equivalence with A and AC function symbols. *Electronic Notes in Theoretical Computer Science*, 332:21–38, 2017.

[AdCSMFR19] M. Ayala-Rincón, W. de Carvalho Segundo, D. Nantes-Sobrinho M. Fernández, and A. C. Rocha-Oliveira. A formalisation of nominal $\alpha$-equivalence with A, C, and AC function symbols. *Theor. Comput. Sci.*, 781:3–23, 2019.

[AFN18] M. Ayala-Rincón, M. Fernández, and D. Nantes-Sobrinho. Fixed-point constraints for nominal equational unification. In *Proc. of the 3rd International Conference on Formal Structures for Computation and Deduction (FSCD)*, volume 108 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:16, 2018.

[AK16] 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.

[ARdCSFNS17] M. Ayala-Rincón, W. de 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 Verlag, 2017.

[ARdCSFNS18] M. Ayala-Rincón, W. de Carvalho Segundo, M. Fernández, and D. Nantes-Sobrinho. *Nominal C-Unification*. In *Proc. of the 27th Int. Symp. Logic-based Program Synthesis and Transformation (LOPSTR 2017), Revised Selected Papers*, volume 10855 of *Lecture Notes in Computer Science*, pages 235–251. Springer, 2018.

[BKN87] D. Benanav, D. Kapur, and P. Narendran. Complexity of Matching Problems. *J. of Sym. Computation*, 3(1/2):203–216, 1987.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and* All That. Cambridge University Press, 1998.

[BS94] F. Baader and J. H. Siekmann. Unification theory. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume2, Deduction Methodologies*, pages 41–126. Oxford University Press, 1994.

[BSN+01] Franz Baader, Wayne Snyder, Paliath Narendran, Manfred Schmidt-Schauß, and Klaus U. Schulz. Unification theory. In *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001.

[Cal10] C. F. Calvès. *Complexity and implementation of nominal algorithms*. PhD Thesis, King's College London, 2010.

[Cal13] C. F. Calvès. Unifying Nominal Unification. In *24th International Conference on Rewriting Techniques and Applications (RTA)*, volume 21 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 143–157, 2013.

[CF08] C. F. Calvès and M. Fernández. A Polynomial Nominal Unification Algorithm. *Theor. Comput. Sci.*, 403(2-3):285–306, 2008.

[Che05] J. Cheney. A Simpler Proof Theory for Nominal Logic. In *Proc. of the 8th Int. Conference on the Foundations of Software Science and Computational Structures (FOSSACS)*, volume 3441 of *Lecture Notes in Computer Science*, pages 379–394. Springer Verlag, 2005.

[Che10] J. Cheney. *Equivariant unification*. *Journal of Automated Reasoning*, 45(3):267–300, 2010.

[CM17] J. Cheney and A. Momigliano. $\alpha$check: A mechanized metatheory model checker. *Theory and Practice of Logic Programming (TPLP)*, 17(3):311–352, 2017.

[Com93] H. Comon. Complete Axiomatizations of Some Quotient Term Algebras. *Theor. Comput. Sci.*, 118(2):167–191, 1993.

[CU08] J. Cheney and C. Urban. Nominal Logic Programming. *ACM Transactions on Programming Languages*, 30(5):1–47, 2008.

[dCS19] W. de Carvalho Segundo. *Nominal Equational Problems Modulo Associativity, Commutativity and Associativity-Commutativity*. PhD thesis, Universidade de Brasília, Brazil, 2019.

[Fag87] F. Fages. Associative-Commutative Unification. *J. of Sym. Computation*, 3:257–275, 1987.

[FG07] M. Fernández and M. J. Gabbay. *Nominal Rewriting*. *Information and Computation*, 205(6):917–965, 2007.

[FGM04] 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.

[Gab00] M. J. Gabbay. *A Theory of Inductive Definitions With $\alpha$-equivalence*. PhD thesis, DPMMS and Trinity College, University of Cambridge, 2000.

[GM08] M. J. Gabbay and A. Mathijssen. Capture-avoiding substitution as a nominal algebra. *Formal Aspects of Computing*, 20(4-5):451–479, 2008.

[GP02] 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.

[Hun80] T. W. Hungerford. *Algebra*. Number 73 in Graduate Texts in Mathematics. Springer Verlag, second edition, 1980.

[LV08] J. Levy and M. Villaret. Nominal Unification from a Higher-Order Perspective. In *Proceedings 19th International Conference on Rewriting Techniques and Applications, RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2008.

[LV10] J. Levy and M. Villaret. *An Efficient Nominal Unification Algorithm*. In *Proc. of the 21st Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 209–226, 2010.

[Pit03] A. M. Pitts. Nominal Logic, a First Order Theory of Names and Binding. *Information and Computation*, 186(2):165–193, 2003.

[Pit13] A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, 2013.

[Sch89] M. Schmidt-Schauß. Unification in Permutative Equational Theories is Undecidable. *J. of Sym. Computation*, 8(4):415–421, 1989.

[Sie90] J. H. Siekmann. Unification theory. *Decision Support Systems*, 6(4):315–337, 1990.

[SKLV17] M. Schmidt-Schauß, T. Kutsia, J. Levy, and M. Villaret. *Nominal Unification of Higher Order Expressions with Recursive Let*. In *Proc. of the 26th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2016), Revised Selected Papers*, volume 10184 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2017.

[Sti81] M. E. Stickel. A Unification Algorithm for Associative-Commutative Functions. *J. of the ACM*, 28(3):423–434, 1981.

[UPG04] C. Urban, A. M. Pitts, and M. J. Gabbay. *Nominal Unification*. *Theor. Comput. Sci.*, 323(1-3):473–497, 2004.