Electronic Notes in Theoretical Informatics And Computer Science

ENTICS
https://entics.episciences.org

Volume 4
Proceedings of MFPS 2024

A Strong Nominal Semantics for Fixed-point Constraints

Ali K. Caires-Santos [a,1] Maribel Fernández[b,2] Daniele Nantes-Sobrinho[a,c,3]

[a] *Department of Mathematics, University of Brasília, Brasília, Brazil*
[b] *Department of Informatics, King's College London, London, UK*
[c] *Department of Computing, Imperial College London, London, UK*

Abstract

Nominal algebra includes α-equality and freshness constraints on nominal terms endowed with a nominal set semantics that facilitates reasoning about languages with binders. Nominal unification is decidable and unitary, however, its extension with equational axioms such as Commutativity (which is finitary) is no longer finitary unless permutation fixed-point constraints are used. In this paper, we extend the notion of nominal algebra by introducing fixed-point constraints and provide a sound semantics using strong nominal sets. We show, by providing a counter-example, that the class of nominal sets is not a sound denotation for this extended nominal algebra. To recover soundness we propose two different formulations of nominal algebra, one obtained by restricting to a class of fixed-point contexts that are in direct correspondence with freshness contexts and another obtained by using a different set of derivation rules.

Keywords: Programming languages, semantics, binding, nominal algebra

1 Introduction

Nominal algebra [16,15] provides a sophisticated framework to interpret and reason algebraically about systems with binding. The term language of nominal algebra is an extension of first-order terms, called nominal terms [22]. Nominal syntax includes *names* (representing object-level variables, or *atoms*), along with meta-level variables (called just *variables*) and constructs to specify binding (nominal *abstraction*). The α-equivalence relation between nominal terms is defined using name *permutations* and an auxiliary *freshness* relation.

Gabbay and Mathijssen [16] define a sound and complete proof system for nominal algebra, where derivation rules are subject to freshness conditions $a\#t$ (read "a is fresh for t" [22]). Nominal algebra with freshness constraints has a semantics in the class of nominal sets (sets with a finitely supported permutation action) which have a rich structure to reason about names, the permutation of names, and name binding. This approach to the specification of syntax with binding has many advantages: Matching and unification modulo α-equivalence are decidable [22] and moreover there are efficient algorithms to compute unique most

⋆ Partially supported by Brazilian FAP-DF Project DE 00193.00001175/2021-11, Brazilian CNPq Project Universal 409003/2021-2
[1] Email: A.K.C.R.Santos@mat.unb.br
[2] Email: maribel.fernandez@kcl.ac.uk
[3] Email: d.nantes-sobrinho@imperial.ac.uk

Published December 15, 2024 Proceedings Available Online at https://doi.org/10.46298/entics.proceedings.mfps40
10.46298/entics.14777

general unifiers (pairs of a freshness context and a substitution) for solvable unification problems [8,19]. These algorithms have been used to implement rewriting, functional languages and logic programming languages (see, e.g., [14,21,9]). Disunification problems are also decidable [5], as well as the more general class of equational problems [6,3].

Since applications in theorem proving and rewrite-based programming languages often require the use of associative-commutative operators, nominal unification algorithms have been extended to deal with equational axioms such as Associativity (A) and Commutativity (C) [7]. C-unification and AC-unification are decidable and finitary, however, although nominal C-unification and nominal AC-unification are decidable, they are not finitary when solutions are expressed using substitutions and freshness constraints. To overcome this limitation, fixed-point constraints $\pi \curlywedge t$ (read "the name permutation π fixes t") should be used: indeed, nominal unification modulo C and AC theories is finitary if solutions are expressed using substitutions, freshness and fixed-point constraints; moreover, freshness constraints can in turn be expressed using fixed-point constraints [4].

The relation between freshness and fixed-point equations was initially proposed by Pitts [1], by the equivalence $a\#x \iff И c.(a\ c) \cdot x = x$, where a and c are atoms and x is an element of a nominal set. The equivalence uses the И ('new') quantifier that quantifies over new/fresh names. Thus, the equivalence means that a is fresh for x if and only if swapping a for any new name c in x does not alter x, in other words, the action of the permutation $(a\ c)$ fixes x. The fixed-point constraints $\pi \curlywedge t$ are more general, since π is not restricted to be of the form $(a\ c)$ for a new name c. For example, depending on the chosen model, a fixed-point constraint $(a\ b) \curlywedge X$ does not necessarily imply $a\#X$. Thus, the semantics of general fixed-point constraints differs from the semantics of freshness constraints and it needs to be determined so that the meaning of the developments in nominal techniques using fixed-point constraints can be established.

In this paper we address this gap. We consider nominal algebras that use fixed-point constraints instead of freshness constraints. Initially, the semantics for nominal algebras with fixed-point constraints was expected to use nominal sets as carriers. However, through the analysis presented in this paper, we show that this assumption is not valid by providing counterexamples. We also show that soundness can be recovered by using a subclass of the class of nominal sets, called strong nominal sets, or by strengthening the derivation rules using a restricted class of fixed-point contexts and the И quantifier from nominal logic.

Summarising, our main contributions are:

(i) A novel semantic interpretation of general fixed-point constraints.

(ii) The definition of a notion of *strong nominal algebra* as well as notions of strong axioms and theories.

(iii) A counter-example showing that the class of nominal sets is not a sound denotation for nominal theories using general fixed-point constraints.

(iv) A proof that strong nominal sets are a sound denotation for nominal theories with fixed-point constraints.

(v) An alternative formulation of a nominal theory with fixed-point constraints, whose denotational semantics is the whole class of nominal sets.

(vi) An analysis of the correspondences between approaches with freshness and fixed-point constraints.

The above results open the way for a generalisation of the algorithms used to check nominal equational problems to the case where the signature includes associative and commutative operators. Recall that equational problems are first-order formulas where the only predicate is equality. They have numerous applications in programming and theorem proving (see, e.g., [11,10,12,13,18]). Extensions to nominal syntax with α-equality and freshness constraints have already been studied [6], but in order to consider syntax modulo equational theories and preserve the properties of the algorithms (in particular to preserve finitary unification) fixed-point constraints are needed, together with a corresponding notion of nominal algebra that can provide sound denotations for the problems. In future work, we will develop algorithms to solve nominal equational problems modulo associative and commutative axioms.

Organisation.
Section 2 contains the basic definitions and notations used throughout the paper. Section 3 establishes the semantics of fixed-point constraints, as well as the failure of soundness for standard derivation rules. Section 4 characterises a class of models for which soundness holds. Section 5 discusses alternative approaches to recover soundness. Applications of the results are discussed in Section 6. Section 7 concludes the paper.

2 Preliminaries

Syntax.
Fix disjoint countably infinite collections $\mathbb{A} = \{a, b, c, \ldots\}$ of *atoms* and $\mathbb{V} = \{X, Y, Z, \ldots\}$ of *unknowns* or *variables*. We consider a *signature* Σ, finite set of *term-formers* ($\mathtt{f}, \mathtt{g}, \ldots$) - disjoint from $\mathbb{A}$ and $\mathbb{V}$ - such that each $\mathtt{f} \in \Sigma$ has a fixed arity, say n, which is a non-negative integer; we write $\mathtt{f} : n$ to denote that $\mathtt{f}$ has arity n. Unless stated otherwise, we follow Gabbay's *permutative convention* [17]: atoms a, b range permutatively over $\mathbb{A}$ therefore saying two atoms a and b are distinct is unnecessary. We use *permutations* to be able to rename atoms, for the purpose of α-conversion. A *finite* permutation π of atoms is a bijection $\mathbb{A} \to \mathbb{A}$ such that the set $\mathtt{dom}(\pi) := \{a \in \mathbb{A} \mid \pi(a) \neq a\}$ is finite; we say that π has *finite domain/support*. Write $\mathtt{id}$ for the *identity permutation*, $\pi \circ \pi'$ for the *composition* of permutations π and π', and π^{-1} for the *inverse* of π. This makes permutations into a group; we will write $\mathtt{Perm}(\mathbb{A})$ to denote the group of finite permutations of atoms.

The syntax of *nominal terms* is defined inductively by the following grammar:

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

where a is an *atom*, $[a]t$ denotes the *abstraction* of the atom a over the term t, $\mathtt{f}(t_1, \ldots, t_n)$ is the *application* of the term-former $\mathtt{f} : n$ to a tuple $(t_1, \ldots, t_n)$ and $\pi \cdot X$ is a *suspension*, where π is an atom permutation. Intuitively, $\pi \cdot X$ denotes that X will get substituted for a term and then π will permute the atoms in that term accordingly (i.e., the permutation π is suspended). Suspensions of the form $\mathtt{id} \cdot X$ will be represented simply by X. The set of all nominal terms formed from a signature Σ will be denoted by $\mathbb{F}(\Sigma, \mathbb{A}, \mathbb{V})$ or simply $\mathbb{F}(\Sigma, \mathbb{V})$ since $\mathbb{A}$ will be fixed.

Abstractions will be used to represent binding operators. A fundamental example is the lambda operator from the λ-calculus. Terms in the λ-calculus will be represented using the signature $\Sigma = \{\mathtt{lam} : 1, \mathtt{app} : 2\}$. If one consider λ-variables as atoms, λ-terms can be inductively generated by the grammar:

$$e ::= a \mid \mathtt{app}(e, e) \mid \mathtt{lam}([a]e)$$

To sugar the notation, we write $\mathtt{app}(s, t)$ as $(s\ t)$ and $\mathtt{lam}([a]s)$ as $\lambda a.s$.

The lambda operator is just one example amongst many others. We can also use abstractions to represent various other binding operators, such as the existential ($\exists$) and universal ($\forall$) quantifiers of first-order logic, or the (ν) operator in the π-calculus.

We say that a nominal term is *ground* (in a fixed signature Σ) when it does not mention unknowns (and mentions only term-formers in Σ). These are inductively defined by the grammar:

$$g, h ::= a \mid [a]g \mid \mathtt{f}(g_1, \ldots, g_n)$$

where here $\mathtt{f}$ ranges over elements of Σ. The set of all ground terms will be denoted by $\mathbb{F}(\Sigma)$. We define the set of *free names* of a ground term g inductively by $\mathtt{fn}(a) := \{a\}, \mathtt{fn}(\mathtt{f}(g_1, \ldots, g_n)) = \bigcup_{i=1}^{n} \mathtt{fn}(g_i)$, and $\mathtt{fn}([a]g) := \mathtt{fn}(g) \setminus \{a\}$.

Write $t \equiv u$ for *syntactic identity* of terms. We define $a \in t$ (reads: "a occurs in t") inductively by:

$$\frac{}{a \in a} \qquad \frac{}{a \in [a]t} \qquad \frac{a \in t}{a \in [b]t} \qquad \frac{a \in \mathtt{dom}(\pi)}{a \in \pi \cdot X} \qquad \frac{a \in t_i \quad (1 \leq i \leq n)}{a \in \mathtt{f}(t_1, \ldots, t_n)}\ .$$

When $a \in t$ does not hold, we write $a \notin t$ and say that "a does not occur in t". Furthermore, we define $\mathtt{Atm}(t) := \{a \mid a \in t\}$, the set of atoms that occur in t.

We define $X \in t$ (reads: "X occurs in t") inductively by:

$$\frac{}{X \in \pi \cdot X} \qquad \frac{X \in t}{X \in [b]t} \qquad \frac{X \in t_i \quad (1 \leq i \leq n)}{X \in \mathtt{f}(t_1, \ldots, t_n)}\,.$$

When $X \in t$ does not hold, we write $X \notin t$ and say that "X does not occur in t". Furthermore, we define $\mathtt{Var}(t) := \{X \mid X \in t\}$, the set of variables that occur in t. Terms whose set of variables is empty, that is, there is no occurrence of variables in their structures, are *ground terms*.

Write $(a\ b)$ for the permutation that *swaps* a for b, i.e., it maps a to b, b to a and all other c to themselves, take $(a\ a) = \mathtt{id}$. To swap atoms a and b in a term t, we use the swapping $(a\ b)$ and write $(a\ b) \cdot t$ to denote the action of replacing a for b and vice-versa in t. An atom permutation is built as a list of swappings. Given two permutations π and ρ, the permutation $\pi^\rho = \rho \circ \pi \circ \rho^{-1}$ denotes the *conjugate* of π with respect to ρ. *Permutation action* on terms is given by: $\pi \cdot a = \pi(a), \pi \cdot (\pi' \cdot X) = (\pi \circ \pi') \cdot X, \pi \cdot ([a]t) = [\pi(a)](\pi \cdot t)$, and $\pi \cdot \mathtt{f}(t_1, \ldots, t_n) = \mathtt{f}(\pi \cdot t_1, \ldots \pi \cdot t_n)$.

Substitutions, denoted by σ, are finite mappings from variables to terms, which extend homomorphically over terms. Atoms can be abstracted by an abstraction operator as in $[a]t$ but cannot be instantiated by a substitution, while unknowns cannot be abstracted but can be instantiated by a substitution.

Constraints, Judgements and Theories.

Differently from the standard nominal approach [1,17,16], we do not use freshness constraints [4], instead we consider the following two kinds of constraints:

- A *fixed-point constraint* is a pair $\pi \curlywedge t$ of a permutation π and a nominal term t. A fixed-point constraint $\pi \curlywedge X$ is called *primitive*. Write Υ and Ψ for finite sets of primitive fixed-point constraints and call them *fixed-point contexts*. $\Upsilon|_X$ denotes the restriction of Υ to X and $\mathtt{perm}(\Upsilon|_X)$ is the set of permutations such that $\pi \curlywedge X$ is in Υ.

  Intuitively, $\pi \curlywedge t$ means that t is a fixed-point of π (i.e., the permutation π fixes t): $\pi \cdot t = t$.
- An *α-equality constraint* (or just equality) is a pair $t = u$ where t and u are nominal terms.

  Equality constraints will be used to state that two terms are *provably equal*.

Accordingly, we consider two judgement forms:

- A *fixed-point judgement form* $\Upsilon \vdash \pi \curlywedge t$ is a pair of a fixed-point context Υ and a fixed-point constraint $\pi \curlywedge t$.
- An *equality judgement form* $\Upsilon \vdash t = u$ is a pair of a fixed-point context Υ and an equality constraint $t = u$.

We may write $\emptyset \vdash \pi \curlywedge t$ as $\vdash \pi \curlywedge t$. Similarly for $\emptyset \vdash t = u$.

Definition 2.1 [Theory] A *theory* $\mathtt{T} = (\Sigma, Ax)$ consists of a signature Σ and a (possibly) infinite set of equality judgement forms Ax, known as the *axioms*.

Example 2.2 $\mathsf{CORE}_{\curlywedge}$ represents a theory with no axioms. More specifically, for any signature Σ, there exists one such theory. Hence, $\mathsf{CORE}_{\curlywedge}^{\Sigma} = (\Sigma, \emptyset)$.

Derivation rules for fixed-point and α-equality judgements are given in Figure 1. Rules ($\curlywedge\mathbf{a}$) and ($\curlywedge\mathtt{f}$) are straightforward. In the rule ($\curlywedge\mathbf{var}$), the condition $\mathtt{dom}(\pi^{\pi'^{-1}}) \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon|_X))$ means that the atoms affected by the permutation $\pi^{\pi'^{-1}}$ are in the domain of the permutations that fix X. The rule ($\curlywedge\mathbf{abs}$) states that $[a]t$ is a fixed-point of π if π fixes $(a\ c_1) \cdot t$, where c_1 is a fresh atom for t (the latter is indicated by the fact that $(c_1\ c_2)$ fixes t). Rules (**refl**), (**symm**) and (**tran**) ensure that equality is an equivalence relation, and the rules (**cong**[]) and (**congf**) ensure that it is a congruence. The ($\mathbf{ax}_{\Upsilon' \vdash \mathbf{t=u}}$) rule instantiates axioms

[4] Pairs of the form $a\#t$ meaning that "a does not occur free in t".

$$\frac{\pi(a) = a}{\Upsilon \vdash \pi \between a}\ (\between\mathbf{a}) \qquad \frac{\Upsilon \vdash \pi \between t_1 \quad \ldots \quad \Upsilon \vdash \pi \between t_n}{\Upsilon \vdash \pi \between \mathtt{f}(t_1, \ldots, t_n)}\ (\between\mathtt{f})$$

$$\frac{\mathtt{dom}(\pi^{\pi'^{-1}}) \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon \vdash \pi \between \pi' \cdot X}\ (\between\mathbf{var}) \qquad \frac{\Upsilon, \overline{(c_1\ c_2) \between \mathtt{Var}(t)} \vdash \pi \between (a\ c_1) \cdot t}{\Upsilon \vdash \pi \between [a]t}\ (\between\mathbf{abs})$$

$$\frac{}{\Upsilon \vdash t = t}\ (\mathbf{refl}) \qquad \frac{\Upsilon \vdash t = u}{\Upsilon \vdash u = t}\ (\mathbf{symm}) \qquad \frac{\Upsilon \vdash t = u \qquad \Upsilon \vdash u = v}{\Upsilon \vdash t = v}\ (\mathbf{tran})$$

$$\frac{\Upsilon \vdash (\pi \cdot \Upsilon')\sigma}{\Upsilon \vdash \pi \cdot t\sigma = \pi \cdot u\sigma}\ (\mathbf{ax}_{\Upsilon' \vdash \mathbf{t=u}}) \qquad \frac{\Upsilon \vdash t = u}{\Upsilon \vdash [a]t = [a]u}\ (\mathbf{cong}[]) \qquad \frac{\Upsilon \vdash t = u}{\Upsilon \vdash \mathtt{f}(\ldots, t, \ldots) = \mathtt{f}(\ldots, u, \ldots)}\ (\mathbf{congf})$$

$$\frac{\Upsilon, \pi \between X \vdash t = u \quad (\mathtt{dom}(\pi) \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon|_X)))}{\Upsilon \vdash t = u}\ (\mathbf{fr})$$

$$\frac{\Upsilon, \overline{(c_1\ c_2) \between \mathtt{Var}(t)} \vdash (a\ c_1) \between t \qquad \Upsilon, \overline{(d_1\ d_2) \between \mathtt{Var}(t)} \vdash (b\ d_1) \between t}{\Upsilon \vdash (a\ b) \cdot t = t}\ (\mathbf{perm})$$

Fig. 1. Derivation rules; c_1, c_2, d_1, d_2 are fresh names. Also, $\overline{\pi \between \mathtt{Var}(t)} = \{\pi \between X \mid X \in \mathtt{Var}(t)\}$.

in derivations. In rule (**ax**) we use the notation $(\pi \cdot \Upsilon')\sigma = \{\pi'^{\pi} \between \pi \cdot X\sigma \mid \pi' \between X \in \Upsilon'\}$. The (**perm**) rule states that if swapping a with a new name c_1 fixes t and similarly swapping b with a new name d_1 fixes t, then swapping a and b does not affect t. Finally, (**fr**) (read top-down) is a strengthening rule. Note that the (**perm**) rule together with the ($\between$) rules implements α-equivalence. For example, we can prove $\vdash [a]a = [b]b$ as follows:

$$\dfrac{\dfrac{\dfrac{(b\ c_4) \cdot b = c_4 \quad \text{and} \quad (a\ c_1)(c_4) = c_4}{\vdash (a\ c_1) \between (b\ c_4) \cdot b}(\between\mathbf{a})}{\vdash (a\ c_1) \between [b]b}(\between\mathbf{abs}) \qquad \dfrac{\dfrac{(b\ c_3) \cdot b = c_3 \quad \text{and} \quad (b\ c_2)(c_3) = c_3}{\vdash (b\ c_2) \between (b\ c_3) \cdot b}(\between\mathbf{a})}{\vdash (b\ c_2) \between [b]b}(\between\mathbf{abs})}{\vdash (a\ b) \cdot [b]b = [b]b}(\mathbf{perm})$$

In the following, $\Upsilon \vdash \pi \between t$ denotes that $\pi \between t$ is derivable using at most the elements of Υ as assumptions. We write $\Upsilon \nvdash \pi \between t$ when $\Upsilon \vdash \pi \between t$ is not derivable. Similarly, for equality judgments we write $\Upsilon \vdash_{\mathtt{T}} t = u$ if $t = u$ can be derived i) using assumptions from Υ; ii) for each occurrence of $(\mathbf{ax}_{\Upsilon' \vdash t=u})$ used in the derivation, $(\Upsilon' \vdash t = u) \in Ax$; iii) only terms in the signature Σ are mentioned in the derivation.

2.1 Nominal Sets

We recall some basic definitions on nominal sets [1].

A $\mathtt{Perm}(\mathbb{A})$-*set*, denoted by $\mathscr{X}$, is a pair $(|\mathscr{X}|, \cdot)$ consisting of an *underlying set* $|\mathscr{X}|$ and a *permutation action* $\cdot$, which is a group action on $|\mathscr{X}|$, i.e., an operation $\cdot : \mathtt{Perm}(\mathbb{A}) \times |\mathscr{X}| \to |\mathscr{X}|$ such that $\mathtt{id} \cdot x = x$ and $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$, for every $x \in |\mathscr{X}|$ and $\pi, \pi' \in \mathtt{Perm}(\mathbb{A})$. Sometimes, we will write $\pi \cdot_{\mathscr{X}} x$, when we want to make $\mathscr{X}$ clear.

For $B \subseteq \mathbb{A}$ write $\mathtt{pfix}(B) = \{\pi \in \mathtt{Perm}(\mathbb{A}) \mid \forall a \in B.\ \pi(a) = a\}$, that is, $\mathtt{pfix}(B)$ is the set of permutations that fix pointwise the elements of B. A set of atomic names $B \subseteq \mathbb{A}$ *supports* an element $x \in |\mathscr{X}|$ when for all permutations $\pi \in \mathtt{Perm}(\mathbb{A})$, $\pi \in \mathtt{pfix}(B) \Longrightarrow \pi \cdot x = x$. Additionally, we say that B *strongly supports* $x \in |\mathscr{X}|$ if for all permutations $\pi \in \mathtt{Perm}(\mathbb{A})$, $\pi \in \mathtt{pfix}(B) \Longleftrightarrow \pi \cdot x = x$.

A *nominal set* is a $\mathtt{Perm}(\mathbb{A})$-set $\mathscr{X}$ all of whose elements are finitely supported. $\mathscr{X}, \mathscr{Y}, \mathscr{Z}$ will range over nominal sets. A nominal set is *strong* if every element is strongly supported by a finite set [5]. The *support*

[5] The class of strong nominal sets is a subclass of the class of nominal sets

of an element $x \in |\mathscr{X}|$ of a nominal set $\mathscr{X}$ is defined as $\mathtt{supp}(x) = \bigcap\{B \mid B \text{ is finite and supports } x\}$. This implies that $\mathtt{supp}(x)$ is the *least* finite support of x. In the case where $\mathscr{X}$ is strong, if x is strongly supported by a finite set $B \subseteq \mathbb{A}$, then $B = \mathtt{supp}(x)$.

For any nominal sets $\mathscr{X}, \mathscr{Y}$, call a map $f : |\mathscr{X}| \to |\mathscr{Y}|$ *equivariant* when $\pi \cdot f(x) = f(\pi \cdot x)$ for all $\pi \in \mathtt{Perm}(\mathbb{A})$ and $x \in |\mathscr{X}|$. In this case we write $f : \mathscr{X} \to \mathscr{Y}$. For instance, any constant map is easily an equivariant map.

Example 2.3 [Some simple nominal sets]

(i) The $\mathtt{Perm}(\mathbb{A})$-set $(\mathbb{A}, \cdot)$ with the action $\pi \cdot_{\mathbb{A}} a = \pi(a)$ is a nominal set and $\mathtt{supp}(a) = \{a\}$.

(ii) Consider the set $\mathcal{P}_{\mathtt{fin}}(\mathbb{A}) = \{B \subset \mathbb{A} \mid B \text{ is finite}\}$. Then the $\mathtt{Perm}(\mathbb{A})$-set $(\mathcal{P}_{\mathtt{fin}}(\mathbb{A}), \cdot)$ with the action $\pi \cdot_{\mathcal{P}_{\mathtt{fin}}(\mathbb{A})} B = \{\pi \cdot_{\mathbb{A}} a \mid a \in B\}$ is a nominal set and $\mathtt{supp}(B) = B$. Observe that $\mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ is not strong because if we take $B = \{a, b\}$ and $\pi = (a\ b)$, then $\pi \cdot B = B$ but $\pi \notin \mathtt{pfix}(B)$.

(iii) The singleton set $\{\star\}$ equipped with the action $\pi \cdot \star = \star$ is a strong nominal set and $\mathtt{supp}(\star) = \emptyset$.

(iv) The set $\mathbb{A}^* = \bigcup\{a_1 \cdots a_n \mid \forall i, j \in \{1, \ldots, n\}. a_i \in \mathbb{A} \wedge (j \neq i \Rightarrow a_j \neq a_i)\}$, that is, the set of finite words over distinct atoms, is a strong nominal set when equipped with the permutation action given by $\pi \cdot (a_1 \cdots a_n) = \pi(a_1) \cdots \pi(a_n)$, and $\mathtt{supp}(a_1 \cdots a_n) = \{a_1, \ldots, a_n\}$.

(v) $\mathbb{F}(\Sigma)$ with the action $\pi \cdot a \equiv \pi(a), \pi \cdot [a]t \equiv [\pi(a)]\pi \cdot t$ and $\pi \cdot \mathtt{f}(t_1, \ldots, t_n) \equiv (\pi \cdot t_1, \ldots, \pi \cdot t_n)$ forms a nominal set and $\mathtt{supp}(g) = \mathtt{Atm}(g)$ for all $g \in \mathbb{F}(\Sigma)$. The relation $\sim$ defined by $g \sim g'$ iff $\vdash_{\mathtt{T}} g = g'$ is an equivariant equivalence relation. The set $\mathbb{F}(\Sigma)/_\sim$ is a nominal set and $\mathtt{supp}(\overline{g}) = \bigcap\{\mathtt{supp}(g') \mid g' \in \overline{g}\}$ for any $\overline{g} \in \mathbb{F}(\Sigma)/_\sim$ (see Proposition 2.30 [1]). We denote $\mathbb{F}(\Sigma)/_\sim$ by $\overline{\mathbb{F}}(\mathtt{T}, \Sigma)$. In the case where $\mathtt{T} = \emptyset$, the relation $\sim$ is the α-equivalence relation between ground terms and $\mathtt{supp}(\overline{g}) = \mathtt{fn}(g)$ for all $\overline{g} \in \overline{\mathbb{F}}(\emptyset, \Sigma)$.

3 Semantics

In this section we provide a semantics for fixed-point constraints using nominal sets, and denote its semantic interpretation as $\curlywedge_{\mathtt{sem}}$. We then build upon the semantic definition of $\curlywedge$ and define concepts such as (strong) Σ-algebras, valuations, interpretations, models, etc. These are necessary ingredients to determine validity of judgements in terms of $\curlywedge_{\mathtt{sem}}$, and consequently, establish soundness of nominal algebra with fixed-point constraints.

Below we overload the symbol $=$ to denote equality between elements of a nominal set.

3.1 Semantics for $\curlywedge$

The semantics of fixed-point constraints $\curlywedge_{\mathtt{sem}}$ differs from $\#_{\mathtt{sem}}$ and it is defined in terms of fixed-point equations, and not in terms of the support. For an element of a nominal set $x \in \mathscr{X}$, if $a \notin \mathtt{supp}(x)$, we write $a \#_{\mathtt{sem}} x$. We refer to [16] for more details about the semantics of nominal algebra defined by freshness constraints. Below we define $\curlywedge_{\mathtt{sem}}$.

Definition 3.1 Let $\mathscr{X}$ be a nominal set, $x \in |\mathscr{X}|, a \in \mathbb{A}$, and $\pi \in \mathtt{Perm}(\mathbb{A})$. We write $\pi \curlywedge_{\mathtt{sem}} x$ to denote that $\pi \cdot x = x$.

Lemma 3.2 *Let $\mathscr{X}$ be a nominal set and $x \in |\mathscr{X}|$. If $\mathtt{dom}(\pi) \cap \mathtt{supp}(x) = \emptyset$ then $\pi \curlywedge_{\mathtt{sem}} x$.*

Proof. Direct from the definition of support. □ □

However, the other direction does not hold in general: if we take $x = \{a, b\}$ and $\pi = (a\ b)$ as in Example 2.3(ii), $\pi \curlywedge_{\mathtt{sem}} x$ but $\mathtt{dom}(\pi) \cap \mathtt{supp}(x) = \{a, b\}$. The converse does hold for strong nominal sets.

Lemma 3.3 *Let $\mathscr{X}$ be a nominal set and $x \in |\mathscr{X}|$. Then*

$$a \in \mathtt{supp}(x) \iff \{c \mid (a\ c) \curlywedge_{\mathtt{sem}} x\} \quad \textit{is finite}$$

Proof. The left-to-right implication follows by showing that $\{c \mid (a\ c) \curlywedge_{\mathtt{sem}} x\} \subseteq \mathtt{supp}(x)$. The right-to-left implication follows by contradiction and Lemma 3.2. □ □

Theorem 3.4 *Let $\mathscr{X}$ be a strong nominal set, $x \in |\mathscr{X}|$, and $\pi, \gamma_1, \ldots, \gamma_n \in \mathtt{Perm}(\mathbb{A})$. If $\gamma_i \curlywedge_{\mathtt{sem}} x$ for every $i = 1, \ldots, n$, and $\mathtt{dom}(\pi) \subseteq \bigcup_{1 \leq i \leq n} \mathtt{dom}(\gamma_i)$, then $\pi \curlywedge_{\mathtt{sem}} x$.*

Proof. For each $i = 1, \ldots, n$ the following hold:

$$\gamma_i \curlywedge_{\mathtt{sem}} x \iff \gamma_i \cdot x = x \iff \gamma_i \in \mathtt{pfix}(\mathtt{supp}(x)).$$

The first equivalence comes from Definition 3.1 and the second from the fact that $\mathscr{X}$ is a strong nominal set. Then by the inclusion $\mathtt{dom}(\pi) \subseteq \bigcup_{1 \leq i \leq n} \mathtt{dom}(\gamma_i)$, we deduce that $\pi \in \mathtt{pfix}(\mathtt{supp}(x))$ which implies, by definition, that $\pi \curlywedge_{\mathtt{sem}} x$. □ □

Note that Theorem 3.4 is not true in the class of nominal sets. For instance, take $\mathscr{X}$ as $\mathcal{P}_{\mathtt{fin}}(\mathbb{A})$, $x = \{a, b\}$, $\pi = (a\ c)$, $\gamma_1 = (a\ b)$, $\gamma_2 = (c\ d)$. Then, $\gamma_1 \curlywedge_{\mathtt{sem}} x, \gamma_2 \curlywedge_{\mathtt{sem}} x$, and $\mathtt{dom}(\pi) \subseteq \mathtt{dom}(\gamma_1) \cup \mathtt{dom}(\gamma_2)$, but $\pi \not\curlywedge_{\mathtt{sem}} x$.

3.2 Strong Nominal Σ-algebras

We can now define a semantics for nominal algebra with fixed-point constraints in (strong) nominal sets.

Definition 3.5 [(Strong) Σ-algebra] Given a signature Σ, a *(strong) nominal Σ-algebra* $\mathfrak{A}$ consists of:

(i) A (strong) nominal set $\mathscr{A} = (|\mathscr{A}|, \cdot)$ - the domain.

(ii) An equivariant map $\mathtt{atom}^{\mathfrak{A}} : \mathbb{A} \to |\mathscr{A}|$ to interpret atoms; we write the interpretation $\mathtt{atom}(a)$ as $a^{\mathfrak{A}} \in |\mathscr{A}|$.

(iii) An equivariant map $\mathtt{abs}^{\mathfrak{A}} : \mathbb{A} \times |\mathscr{A}| \to |\mathscr{A}|$ such that $\{c \in \mathbb{A} \mid (a\ c) \not\curlywedge_{\mathtt{sem}} \mathtt{abs}^{\mathfrak{A}}(a, x)\}$ is finite, for all $a \in \mathbb{A}$ and $x \in |\mathscr{A}|$ (by Lemma 3.3 this is equivalent to saying that $a \notin \mathtt{supp}(\mathtt{abs}^{\mathfrak{A}}(a, x))$); we use this to interpret abstraction.

(iv) An equivariant map $f^{\mathfrak{A}} : |\mathscr{A}|^n \to |\mathscr{A}|$, for each $\mathtt{f} : n$ in Σ to interpret term-formers.

We use $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$ to denote (strong) Σ-algebras.

Remark 3.6 The standard definition of the condition in item 3 via freshness constraints is $a \#_{\mathtt{sem}} \mathtt{abs}^{\mathfrak{A}}(a, x)$ (see, for instance, [16]). Here, we use the following equivalence (valid in nominal sets):

$$\begin{aligned} a \#_{\mathtt{sem}} \mathtt{abs}^{\mathfrak{A}}(a, x) &\Leftrightarrow a \notin \mathtt{supp}(\mathtt{abs}^{\mathfrak{A}}(a, x)) && \text{(Definition 4.2, [16])} \\ &\Leftrightarrow \{c \in \mathbb{A} \mid (a\ c) \not\curlywedge_{\mathtt{sem}} \mathtt{abs}^{\mathfrak{A}}(a, x)\} \text{ is finite} && \text{(Consequence of Lemma 3.3)} \end{aligned}$$

A *valuation* ς in a (strong) Σ-algebra $\mathfrak{A}$ maps unknowns $X \in \mathbb{V}$ to elements $\varsigma(X) \in |\mathscr{A}|$. Below we define an equivariant function $[\![\cdot]\!]^{\mathfrak{A}}_{\varsigma} : T(\Sigma, \mathbb{A}, \mathbb{V}) \to |\mathscr{A}|$ to interpret nominal terms w.r.t. a valuation ς.

Definition 3.7 [Interpretation] Let $\mathfrak{A}$ be a (strong) Σ-algebra. Suppose that $t \in T(\Sigma, \mathbb{A}, \mathbb{V})$ and consider a valuation ς in $\mathfrak{A}$. The *interpretation* $[\![t]\!]^{\mathfrak{A}}_{\varsigma}$, or just $[\![t]\!]_{\varsigma}$, if $\mathfrak{A}$ is understood, is defined inductively by:

$$[\![a]\!]^{\mathfrak{A}}_{\varsigma} = a^{\mathfrak{A}} \qquad [\![\pi \cdot X]\!]^{\mathfrak{A}}_{\varsigma} = \pi \cdot \varsigma(X)$$

$$[\![\mathtt{f}(t_1, \ldots, t_n)]\!]^{\mathfrak{A}}_{\varsigma} = f^{\mathfrak{A}}([\![t_1]\!]^{\mathfrak{A}}_{\varsigma}, \ldots, [\![t_n]\!]^{\mathfrak{A}}_{\varsigma}) \qquad [\![[a]t]\!]^{\mathfrak{A}}_{\varsigma} = \mathtt{abs}(a^{\mathfrak{A}}, [\![t]\!]^{\mathfrak{A}}_{\varsigma}).$$

Lemma 3.8 *The term-interpretation map $[\![\cdot]\!]^{\mathfrak{A}}_{\varsigma}$ is equivariant, that is, $\pi \cdot [\![t]\!]^{\mathfrak{A}}_{\varsigma} = [\![\pi \cdot t]\!]^{\mathfrak{A}}_{\varsigma}$ for any permutation π and term t.*

Proof. The proof is by induction on the structure of the term t. □ □

Definition 3.9 [Context and Judgement Validity] For any (strong) Σ-algebra $\mathfrak{A}$:

- $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ is *valid* iff $\pi \curlywedge_{\mathtt{sem}} [\![X]\!]^{\mathfrak{A}}_{\varsigma}$, for each $\pi \curlywedge X \in \Upsilon$;
- $[\![\Upsilon \vdash \pi \curlywedge t]\!]^{\mathfrak{A}}_{\varsigma}$ is *valid* iff $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ (valid) implies $\pi \curlywedge_{\mathtt{sem}} [\![t]\!]^{\mathfrak{A}}_{\varsigma}$;
- $[\![\Upsilon \vdash t = u]\!]^{\mathfrak{A}}_{\varsigma}$ is *valid* iff $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ (valid) implies $[\![t]\!]^{\mathfrak{A}}_{\varsigma} = [\![u]\!]^{\mathfrak{A}}_{\varsigma}$.

More generally, $[\![\Upsilon \vdash \pi \curlywedge t]\!]^{\mathfrak{A}}$ is *valid* iff $[\![\Upsilon \vdash \pi \curlywedge t]\!]^{\mathfrak{A}}_{\varsigma}$ is valid for all valuations ς. Similarly for $[\![\Upsilon \vdash t = u]\!]^{\mathfrak{A}}$.

Then a (strong) model of a theory is an interpretation that validates its axioms as follows:

Definition 3.10 [(Strong) Model] Let $\mathtt{T} = (\Sigma, Ax)$ be a theory. A *(strong) model of* $\mathtt{T}$ is a (strong) Σ-algebra $\mathcal{A}$ such that

$$[\![\Upsilon \vdash t = u]\!]^{\mathfrak{A}} \text{ is valid for every axiom } \Upsilon \vdash t = u \text{ in } Ax \text{ and every valuation } \varsigma.$$

We are now ready to define validity with respect to a theory.

Definition 3.11 For any theory $\mathtt{T}$, define *(strong) validity with respect to* $\mathtt{T}$ for judgement forms as follows:

- Write $\Upsilon \vDash_{\mathtt{T}} \pi \curlywedge t$ (resp. $\Upsilon \vDash^{s}_{\mathtt{T}} \pi \curlywedge t$) iff $[\![\Upsilon \vdash \pi \curlywedge t]\!]^{\mathfrak{A}}$ is valid for all models $\mathfrak{A}$ of $\mathtt{T}$ (resp. all strong models of $\mathtt{T}$).
- Write $\Upsilon \vDash_{\mathtt{T}} t = u$ (resp. $\Upsilon \vDash^{s}_{\mathtt{T}} t = u$) iff $[\![\Upsilon \vdash t = u]\!]^{\mathfrak{A}}$ is valid for all models $\mathfrak{A}$ of $\mathtt{T}$ (resp. all strong models of $\mathtt{T}$).

Lemma 3.12 *Define the* singleton interpretation $\mathfrak{S}$ *to have as domain the strong nominal set* $\{\star\}$*, with* $\mathtt{atom}^{\mathfrak{S}}, \mathtt{abs}^{\mathfrak{S}}, f^{\mathfrak{S}}$ *constant functions with value* $\star$*. Then* $\mathfrak{S}$ *is a strong* Σ*-algebra and a model for every theory.*

Proof. To verify that $\mathfrak{S}$ is a model of $\mathtt{T}$, check that $[\![\Upsilon \vdash t = u]\!]^{\mathfrak{S}}_{\varsigma}$ is valid for each axiom $\Upsilon \vdash t = u \in Ax$ and valuation ς (which maps all unknowns to $\star$). If $[\![\Upsilon]\!]^{\mathfrak{S}}_{\varsigma}$ is valid, then $[\![t]\!]^{\mathfrak{S}}_{\varsigma} = [\![u]\!]^{\mathfrak{S}}_{\varsigma} = \star$, confirming $\mathfrak{S}$ as a model. □ □

Remark 3.13 Lemma 3.12 shows that we do not need to require the map $\mathtt{atom}$ to be injective.

3.3 Counter-Example: Soundness Failure

Soundness of a derivation system means that derivability implies validity. More precisely, in a sound system the following should hold:
(Soundness) Suppose $\mathtt{T} = (\Sigma, Ax)$ is a theory.

(i) If $\Upsilon \vdash \pi \curlywedge t$ then $\Upsilon \vDash_{\mathtt{T}} \pi \curlywedge t$.

(ii) If $\Upsilon \vdash_{\mathtt{T}} t = u$ then $\Upsilon \vDash_{\mathtt{T}} t = u$.

Note that by definition, for a judgement to be valid it has to hold in all models $\mathfrak{A}$ of $\mathtt{T}$ for every valuation ς. We present two counter-examples that illustrate two different points where soundness fails.

3.3.1 Rule $\curlywedge$**var**.

The problem arises from the rule ($\curlywedge$**var**):

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

Claim 3.14 *There exist a* Σ*-algebra* $\mathfrak{A}$*, a valuation* ς *and a derivation using rule* ($\curlywedge$**var**) *such that* $\Upsilon \vdash \pi \curlywedge \pi' \cdot X$ *but* $[\![\Upsilon \vdash \pi \curlywedge \pi' \cdot X]\!]^{\mathfrak{A}}_{\varsigma}$ *is not valid.*

First of all, fix an enumeration of $\mathbb{V} = \{X_1, X_2, \ldots\}$ and $\mathbb{A} = \{a_1, a_2, \ldots\}$. Now, let's give the nominal set $\mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ a Σ-algebra structure.

(i) Consider the domain of $\mathfrak{A}$ as the nominal set $\mathscr{A} = (\mathcal{P}_{\mathtt{fin}}(\mathbb{A}), \cdot)$ presented in Example 2.3(ii).

(ii) Define $\mathtt{atom}^{\mathfrak{A}}\colon \mathbb{A} \to \mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ by $\mathtt{atom}^{\mathfrak{A}}(a) = \{a\}$.

(iii) Define $\mathtt{abs}^{\mathfrak{A}} : \mathbb{A} \times \mathcal{P}_{\mathtt{fin}}(\mathbb{A}) \to \mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ by $\mathtt{abs}^{\mathfrak{A}}(a, B) = B \setminus \{a\}$.

(iv) For each term-former $\mathtt{f}\colon n \in \Sigma$, we associate a map $f^{\mathfrak{A}}\colon \mathcal{P}_{\mathtt{fin}}(\mathbb{A})^n \to \mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ defined by $f^{\mathfrak{A}}(B_1, \ldots, B_n) = \bigcap_{i=1}^{n} B_i$.

It is easy to see that the functions $\mathtt{atom}^{\mathfrak{A}}, \mathtt{abs}^{\mathfrak{A}}, f^{\mathfrak{A}}$ are equivariant.

Define the valuation $\varsigma : \mathbb{V} \to \mathcal{P}_{\mathtt{fin}}(\mathbb{A})$ by $\varsigma(X_i) = \{a_i, a_{i+1}\}$. Now, let's build a derivation using the rule ($\curlywedge\mathbf{var}$) and interpret it in $\mathfrak{A}$. Consider $\Upsilon = \{(a_1\ a_2) \curlywedge X_1, (a_3\ a_4) \curlywedge X_1\}$ and write $\gamma_1 = (a_1\ a_2)$ and $\gamma_2 = (a_3\ a_4)$. Take $\pi = (a_1\ a_3)$. Hence

$$\mathtt{dom}(\pi) = \{a_1, a_3\} \subseteq \{a_1, a_2, a_3, a_4\} = \mathtt{dom}(\gamma_1) \cup \mathtt{dom}(\gamma_2).$$

So $\Upsilon \vdash \pi \curlywedge X_1$ is derivable. All we need to do now is to verify the validity of $[\![\Upsilon \vdash \pi \curlywedge X_1]\!]^{\mathfrak{A}}_{\varsigma}$. In order to so, we need to prove that

$$\text{If } [\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma} \text{ is valid then } \pi \curlywedge_{\mathtt{sem}} [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}.$$

First, observe that $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ is in fact valid:

$$\begin{aligned}\gamma_1 \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} &= \gamma_1 \cdot \{a_1, a_2\} = \{a_1, a_2\} = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}\\ \gamma_2 \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} &= \gamma_2 \cdot \{a_1, a_2\} = \{a_1, a_2\} = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}.\end{aligned}$$

However,

$$\begin{aligned}\pi \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} &= \pi \cdot \varsigma(X_1)\\ &= \pi \cdot \{a_1, a_2\}\\ &= \{a_3, a_2\} \neq \{a_1, a_2\} = \varsigma(X_1) = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}.\end{aligned}$$

3.3.2 Non-strong Axioms.
Let $\mathtt{C}$ be the commutativity axiom: $\mathtt{C} = \{\vdash X + Y = Y + X\}$. Consider the Σ-algebra with domain $\overline{\mathbb{F}}(\mathtt{C}, \Sigma)$, that is the set of ground nominal terms quotiented by $\mathtt{C}$ and α-equality. Let $[t]_{\mathtt{C}}$ denote the equivalence class of $t \in \mathbb{F}(\Sigma)$ modulo $\alpha, \mathtt{C}$.

Consider the same permutations γ_1, γ_2, π as above, so again we can derive $\Upsilon \vdash \pi \curlywedge X_1$.

Take $\varsigma(X_1) = [a_1 + a_2]_{\mathtt{C}}$. Then $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ is valid:

$$\gamma_1 \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} = [a_2 + a_1]_{\mathtt{C}} = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} \qquad \gamma_2 \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} = [a_2 + a_1]_{\mathtt{C}} = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}.$$

However, $\pi \not\curlywedge_{\mathtt{sem}} [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}$ since $\pi \cdot [\![X_1]\!]^{\mathfrak{A}}_{\varsigma} = [a_3 + a_2]_{\mathtt{C}} \neq [a_1 + a_2]_{\mathtt{C}} = [\![X_1]\!]^{\mathfrak{A}}_{\varsigma}$.

Remark 3.15 $\overline{\mathbb{F}}(\mathtt{C}, \Sigma)$ is not a strong nominal set: note that $(a\ b) \curlywedge_{\mathtt{sem}} [a+b]_{\mathtt{C}}$, since $(a\ b) \cdot [a+b]_{\mathtt{C}} = [a+b]_{\mathtt{C}}$, but $(a\ b) \not\curlywedge_{\mathtt{sem}} a$, i.e., $(a\ b) \notin \mathtt{pfix}(\mathtt{supp}([a + b]_{\mathtt{C}})) = \mathtt{pfix}(\{a, b\})$.

4 Recovering Soundness: Strong Models

To avoid the problems discussed in the previous section, we now focus on strong models. With these models, it is possible to ensure soundness.

Theorem 4.1 (Soundness for strong Σ-algebras) *Suppose $\mathtt{T} = (\Sigma, Ax)$ is a theory. Then the following hold:*

(i) *If $\Upsilon \vdash_{\mathtt{T}} \pi \curlywedge t$ then $\Upsilon \vDash^{s}_{\mathtt{T}} \pi \curlywedge t$.*

(ii) *If $\Upsilon \vdash_{\mathtt{T}} t = u$ then $\Upsilon \vDash^{s}_{\mathtt{T}} t = u$.*

Proof. The proof is done by induction on the derivation of $\Upsilon \vdash \pi \curlywedge t$, focusing on the last rule applied. Here, we demonstrate the most interesting case, where the last rule is ($\curlywedge$**var**):

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

Suppose that $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ is valid. We need to show that $\pi \cdot [\![\pi' \cdot X]\!]^{\mathfrak{A}}_{\varsigma} = [\![\pi' \cdot X]\!]^{\mathfrak{A}}_{\varsigma}$.

If $\mathtt{perm}(\Upsilon|_X) = \{\gamma_1, \ldots, \gamma_n\}$, then from the inclusion $\mathtt{dom}(\pi^{\pi'^{-1}}) \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon|_X))$ we get $\mathtt{dom}(\pi^{\pi'^{-1}}) \subseteq \bigcup_{i=1}^{n} \mathtt{dom}(\gamma_i)$.

Moreover, the validity of $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ implies that $\gamma_i \curlywedge_{\mathtt{sem}} [\![X]\!]^{\mathfrak{A}}_{\varsigma}$ for every $\gamma_i \curlywedge X \in \Upsilon$. Thus, $\pi^{\pi'^{-1}} \cdot [\![X]\!]^{\mathfrak{A}}_{\varsigma} = [\![X]\!]^{\mathfrak{A}}_{\varsigma}$ by Theorem 3.4. The result follows by noticing that $\pi^{\pi'^{-1}} \cdot [\![X]\!]^{\mathfrak{A}}_{\varsigma} = [\![X]\!]^{\mathfrak{A}}_{\varsigma} \iff \pi \cdot [\![\pi' \cdot X]\!]^{\mathfrak{A}}_{\varsigma} = [\![\pi' \cdot X]\!]^{\mathfrak{A}}_{\varsigma}$.□

4.1 Strong Axioms

Example 3.3.2 demonstrates that there exist theories $\mathtt{T}$ for which $\overline{\mathbb{F}}(\mathtt{T}, \Sigma)$ is not a strong nominal set. This observation rises the following question: are there theories $\mathtt{T}$ for which $\overline{\mathbb{F}}(\mathtt{T}, \Sigma)$ forms a strong nominal set? We propose to characterise such theories by defining a notion of *strong axiom*, and we will refer to them as *strong theories*.

In the definition below, let $<_{lex}$ be the lexicographic ordering of the positions in a term: e.g., $1.2 <_{lex} 2.2$, $1.1 <_{lex} 1.2$.

Given a term t, we can order the occurrences of variables in t according to their position: we write $X <_t Y$ if X occurs in t at a position p and Y occurs in t at a position q such that $p <_{lex} q$.

We say that a term t is well-ordered if $<_t$ is a strict partial order, i.e., there is no $X, Y \in \mathtt{Var}(t)$ such that $X <_t Y$ and $Y <_t X$. We will only consider axioms with well-ordered first-order terms.

Definition 4.2 An axiom $\Upsilon \vdash t = u$ is *strong* if the following hold:

(i) t and u are first-order terms (i.e., they are built using function symbols and variables), and $\Upsilon = \emptyset$;

(ii) u and t are well-ordered;

(iii) the order of the variables that occur in t and in u is compatible: i.e., if $X <_t Y$ then it is not the case that $Y <_u X$.

Note that it could be that the variables X and Y occur in t and not in u, the important fact is that there are no disagreements in the ordering in t and u.

Condition (i) excludes axioms such as $Ax = \{ \ \vdash a = b\}$, $Ax = \{ \ \vdash a + b = b + a\}$ and $Ax = \{ \ \vdash \mathtt{f}([a]X, [b]Y) = \mathtt{g}([b]X, [a]Y)\}$. Condition (ii) excludes axioms such as distributivity $\mathtt{D} = \{ \ \vdash X * (Y + Z) = X * Y + X * Z\}$. Permutative theories, where axioms have the form $\ \vdash \mathtt{f}(X_1, \ldots, X_n) = \mathtt{f}(X_{\rho(1)}, \ldots, X_{\rho(n)})$, for some permutation ρ of $\{1, \ldots, n\}$, do not satisfy the definition either, i.e. permutative axioms are not strong.

Example 4.3 [Strong Axioms] The following axioms (and their combinations) are strong: Associativity $\mathtt{A} = \{ \ \vdash \mathtt{f}(\mathtt{f}(X, Y), Z) = \mathtt{f}(X, \mathtt{f}(Y, Z))\}$. Homomorphism $\mathtt{Hom} = \{ \ \vdash \mathtt{h}(X + Y) = \mathtt{h}(X) + \mathtt{h}(Y)\}$. Idempotency $\mathtt{I} = \{ \ \vdash \mathtt{g}(X, X) = X\}$. Neutral element $\mathtt{N} = \{ \ \vdash X * 0 = 0\}$. Left-/right-projection $\mathtt{Lproj} = \{ \ \vdash \mathtt{pl}(X, Y) = X\}$ and $\mathtt{Rproj} = \{ \ \vdash \mathtt{pr}(X, Y) = Y\}$.

Example 4.4 [Theory ATOM] Consider the axiom $Ax = \{ \ \vdash a = b\}$ that generates the theory ATOM. By our Definition 4.2, this axiom is not strong.

Theorem 4.5 *If* $\mathtt{T}$ *is a strong theory then* $\overline{\mathbb{F}}(\mathtt{T}, \Sigma)$ *is a strong nominal set.*

Proof. First, as observed in Example 2.3(v) the set $\overline{\mathbb{F}}(\mathtt{T}, \Sigma)$ is a nominal set and $\mathtt{supp}(\overline{g}) = \bigcap\{\mathtt{supp}(x) \mid x \in \overline{g}\}$ for all $\overline{g} \in \overline{\mathbb{F}}(\mathtt{T}, \Sigma)$. It remains to prove that $\mathtt{supp}(\overline{g})$ is a strong support, that is, for all permutations π, the following holds

$$\pi \curlywedge_{\mathtt{sem}} \overline{g} \Longrightarrow \pi \in \mathtt{pfix}(\mathtt{supp}(\overline{g})).$$

Assume, by contradiction, that $\pi \curlywedge_{\mathtt{sem}} \overline{g}$ and $\pi \notin \mathtt{pfix}(\mathtt{supp}(\overline{g}))$, that is, there is an atom $a \in \mathtt{supp}(\overline{g})$ such that $\pi(a) \neq a$; let's assume $\pi(a) = b$.

From $\pi \curlywedge_{\mathtt{sem}} \overline{g}$ it follows, by definition, that $\overline{\pi \cdot g} = \pi \cdot \overline{g} = \overline{g}$. Thus, $\vdash_{\mathtt{T}} \pi \cdot g = g$ (see Example 2.3(v)). Now, we proceed by analysing the derivation of $\vdash_{\mathtt{T}} \pi \cdot g = g$:

- If rule **(ax)** is not applied in the derivation, then $\vdash \pi \cdot g = g$. Since the application of a permutation does not change the structure of terms, it must be the case that $\pi \cdot c = c$, for every free atom c of g. This contradicts the assumption above that $\pi(a) = b$. Therefore, $\pi \in \mathtt{pfix}(\mathtt{supp}(\overline{g}))$ and the result follows.
- If rule **(ax)** is applied in the derivation of $\vdash_{\mathtt{T}} \pi \cdot g = g$, assume it is the last step (the reasoning generalises by induction). Suppose the axiom $\vdash t = u \in Ax$ (in $\mathtt{T}$) is used. Since the axiom is strong, the order of the atoms a, b, that are introduced by instantiation of the variables in the axiom, is preserved in both sides. Thus, the order of these atoms is preserved in the derivation of $\vdash_{\mathtt{T}} \pi \cdot g = g$. This contradicts the hypothesis that $\pi(a) = b$: if a occurs before b in an instance of t that is equal to g (in an application of **(ax)**), then b will occur before a, in $\pi \cdot g$, and this contradicts the fact that $\mathtt{T}$ is strong. Therefore, $\pi \in \mathtt{pfix}(\mathtt{supp}(\overline{g}))$, and the result follows.

□

5 Recovering Soundness: Alternative Approaches

In the previous section we proved soundness with respect to strong models (where carriers are strong nominal sets). Here our goal is to recover soundness while using a nominal set semantics. For this, we need to address the mismatch between the derivations and the notion of validity. We can proceed in different ways:

(i) we can change the form of the judgements and adapt the rules to ensure only the ones that are valid in all models can be derived; or

(ii) we do not change the judgements but we change the derivation rules to prevent the derivation of judgements that are not valid in all models.

The challenge with the second approach is to ensure the system is sufficiently powerful to derive all the valid judgements. The challenge with the first is to ensure that we do not lose expressive power. In this section we show how to address these challenges: we define two sound systems for nominal algebra with fixed-point constraints and nominal set semantics, one with similar rules as before but stronger judgements and another with the same judgements as before but stronger rules.

5.1 Using Strong Fixed-Point Contexts

In this section we present an alternative proof system for nominal algebra with fixed-point constraints. It restricts the form of fixed-point contexts in judgements, which will now be called *strong fixed-point contexts* and include a $И$-quantifier [20] to keep track of new atoms in fixed-point constraints.

Definition 5.1 [Strong Fixed-Point Context] A *strong fixed-point context* is a finite set $\Upsilon_{A,C}$ that contains only primitive constraints of the form $Иc.(a\ c) \curlywedge X$ such that $a \in A$ and $c \in C$, for two given disjoint set of atoms A and C. We will move the $И$-quantifiers to the front and write $ИC.\Upsilon_{A,C}$. We will omit the indices A, C when there is no ambiguity. As before, we will use the symbols $\Upsilon, \Psi, \ldots$ to represent strong contexts.

Intuitively, strong fixed-point contexts contain only fixed-point constraints that correspond directly to freshness constraints if we assume that the set C contains new atoms: $Иc.(a\ c) \curlywedge X$ corresponds to $a\#X$.

Definition 5.2 [Strong Judgements] A *strong fixed-point judgement* has the form $И\overline{c}.(\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge t)$ where $\Upsilon_{A,\overline{c_0}}$ is a strong fixed-point context and $\overline{c_0} \subseteq \overline{c}$. Similarly, a strong α-equality judgement has the form $И\overline{c}.(\Upsilon_{A,\overline{c_0}} \vdash s \overset{\curlywedge}{\approx} t)$ where $\Upsilon_{A,\overline{c_0}}$ is a strong fixed-point context and $\overline{c_0} \subseteq \overline{c}$.

$$\frac{\pi(a) = a}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge a}\ (\curlywedge\mathbf{a}) \qquad \frac{\mathtt{dom}(\pi^{\rho^{-1}}) \setminus \overline{c} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge \rho \cdot X}\ (\curlywedge\mathbf{var})$$

$$\frac{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge t_1 \quad \ldots \quad И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge t_n}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge \mathtt{f}(t_1,\ldots,t_n)}\ (\curlywedge\mathtt{f}) \qquad \frac{И\overline{c}, c_1.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge (a\ c_1) \cdot t}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge [a]t}\ (\curlywedge\mathbf{abs})$$

$$\frac{}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash a \mathrel{\curlywedge\!\!\approx} a}\ (\mathrel{\curlywedge\!\!\approx}\mathbf{a}) \qquad \frac{\mathtt{dom}(\rho^{-1} \circ \pi) \setminus \overline{c} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \cdot X \mathrel{\curlywedge\!\!\approx} \rho \cdot X}\ (\mathrel{\curlywedge\!\!\approx}\mathbf{var})$$

$$\frac{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash t_1 \mathrel{\curlywedge\!\!\approx} t'_1 \quad \ldots \quad И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash t_n \mathrel{\curlywedge\!\!\approx} t'_n}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \mathtt{f}(t_1,\ldots,t_n) \mathrel{\curlywedge\!\!\approx} \mathtt{f}(t'_1,\ldots,t'_n)}\ (\mathrel{\curlywedge\!\!\approx}\mathtt{f}) \qquad \frac{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash t \mathrel{\curlywedge\!\!\approx} t'}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash [a]t \mathrel{\curlywedge\!\!\approx} [a]t'}\ (\mathrel{\curlywedge\!\!\approx}[\mathbf{a}])$$

$$\frac{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash s \mathrel{\curlywedge\!\!\approx} (a\ b) \cdot t \qquad И\overline{c}, c_1.\Upsilon_{A,\overline{c_0}} \vdash (a\ c_1) \curlywedge t}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash [a]s \mathrel{\curlywedge\!\!\approx} [b]t}\ (\mathrel{\curlywedge\!\!\approx}\mathbf{ab})$$

Fig. 2. Derivation rules for strong judgements. Here, $\overline{c}$ denotes a list of distinct atoms $c_1,\ldots,c_n$. In all the rules $\overline{c_0} \subseteq \overline{c}$.

$$\frac{}{\Delta \vdash a\#b}(\#a) \qquad \frac{\pi^{-1}(a)\#X \in \Delta}{\Delta \vdash a\#\pi \cdot X}(\#var) \qquad \frac{}{\Delta \vdash a\#[a]t}(\#[a]) \qquad \frac{\Delta \vdash a\#t}{\Delta \vdash a\#[b]t}(\#abs)$$

$$\frac{\Delta \vdash a\#t_1 \quad \ldots \quad \Delta \vdash a\#t_n}{\Delta \vdash a\#\mathtt{f}(t_1,\ldots,t_n)}(\#\mathtt{f}) \qquad \frac{}{\Delta \vdash a \approx a}(\approx a) \qquad \frac{ds(\pi,\pi')\#X \subseteq \Delta}{\Delta \vdash \pi \cdot X \approx \pi' \cdot X}(\approx var) \qquad \frac{\Delta \vdash s \approx t}{\Delta \vdash [a]s \approx [a]t}(\approx [a])$$

$$\frac{\Delta \vdash s \approx (a\ b) \cdot t \qquad \Delta \vdash a\#t}{\Delta \vdash [a]s \approx [b]t}(\approx ab) \qquad \frac{\Delta \vdash s_1 \approx t_1 \quad \ldots \quad \Delta \vdash s_n \approx t_n}{\Delta \vdash \mathtt{f}(s_1,\ldots,s_n) \approx \mathtt{f}(t_1,\ldots,t_n)}(\approx \mathtt{f})$$

Fig. 3. Standard derivation rules for freshness and α-equivalence. Here, $\mathtt{ds}(\pi,\pi') = \{a \in \mathbb{A} \mid \pi(a) \neq \pi'(a)\}$.

Note that the И quantifiers at the front of the judgement quantify the constraints in the context: for example in the judgement $Иc_1, c_2.(a\ c_1) \curlywedge X, (a\ c_2) \curlywedge X \vdash s \mathrel{\curlywedge\!\!\approx} t$ we have $Иc_1.(a\ c_1) \curlywedge X$ in the context.

The alternative proof system is given in Figure 2. It is a modified version of a system introduced in [4], that also used И in the derivable constraints (on the rhs of the $\vdash$), but did not use strong fixed-point contexts. In Figure 2 we adapt the notations $\mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c}}|_X)) = \{a \mid (a\ c) \curlywedge X \in \Upsilon_{A,\overline{c}}\}$.

The following correctness result states that, under strong fixed-point judgements, the fixed-point constraint $\curlywedge$ is still a fixed-point relation.

Theorem 5.3 (Correctness) *$И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \curlywedge t$ iff $И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \cdot t \mathrel{\curlywedge\!\!\approx} t$, where $\overline{c_0} \subseteq \overline{c}$.*

Proof. The proof follows by induction on the derivations. □ □

5.1.1 From # to $\curlywedge$ and back.

We recall some basic definitions of nominal algebra with freshness constraints. A freshness constraint is a pair of the form $a\#t$ where a is an atom and t is a nominal term, and it denotes the fact that 'a is fresh for t', that is, if a occurs in t it must occur abstracted. A freshness context (denoted Δ, ∇) is a finite set of primitive freshness constraints of the form $a\#X$. An α-equality constraint in the nominal algebra with # is denoted $s \approx t$, where s and t are nominal terms. A judgement has the form $\Delta \vdash a\#t$ or $\Delta \vdash s \approx t$. The standard proof system for deriving freshness judgements is defined by the rules in Figure 3.

Recall that the notion of an atom a being fresh for an element x in nominal set is defined using the

И-quantifier and a fixed-point equation as follows [1]:

$$a\#x \iff \text{И}c.(a\ c)\cdot x = x.$$

We explore this relationship and define mappings that translate strong judgements using $\curlywedge$ and $\overset{\curlywedge}{\approx}$ into judgments using $\#$ and $\approx$, and vice-versa. Below we denote by $\mathfrak{F}_{\#}$ the family of (primitive) freshness constraints, and by $\mathfrak{F}_{\curlywedge}$ the family of primitive strong fixed-point constraints. The mapping $[\cdot]_{\curlywedge}$ associates each primitive freshness constraint with a primitive strong fixed-point constraint; it extends to freshness contexts in a natural way

$$\begin{aligned}[\cdot]_{\curlywedge} &: \mathfrak{F}_{\#} \to \mathfrak{F}_{\curlywedge}\\ a\#X &\mapsto \text{И}c_a.(a\ c_a) \curlywedge X\end{aligned}$$

We denote by $[\Delta]_{\curlywedge}$ the image of Δ under $[\cdot]_{\curlywedge}$. The mapping $[\cdot]_{\#}$ associates each primitive strong fixed-point constraint with a freshness constraint; it extends to fixed-point contexts in a natural way.

$$\begin{aligned}[\cdot]_{\#} &: \mathfrak{F}_{\curlywedge} \to \mathfrak{F}_{\#}\\ \text{И}c.(a\ c) \curlywedge X &\mapsto a\#X.\end{aligned}$$

Our translations map strong fixed-point judgments into freshness judgements. The next result establishes that every derivable freshness judgement can be mapped, via $[\cdot]_{\curlywedge}$ to a strong fixed-point constraint.

Theorem 5.4 (From and to $[\cdot]_{\curlywedge}$) *The following hold, for some $\overline{c}$ (possibly empty):*

(i) $\Delta \vdash a\#t \iff \text{И}\overline{c}, c'.[\Delta]_{\curlywedge} \vdash (a\ c') \curlywedge t.$

(ii) $\Delta \vdash s \approx t \iff \text{И}\overline{c}.[\Delta]_{\curlywedge} \vdash s \overset{\curlywedge}{\approx} t.$

Proof. The proof is by induction on the last rule applied in the derivation. We will prove only the interesting case involving the abstraction rule for item (i). The proof for item (ii) is analogous.

($\Rightarrow$) : If the last rule applied is ($\#abs$), then $t \equiv [b]t'$ and there is a derivation

$$\frac{\Delta \vdash a\#t'}{\Delta \vdash a\#[b]t'}\ (\#abs)$$

By induction, there exists a proof $\text{И}\overline{c}, c'.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge t'$. Note now that for any fresh name c_1 for the derivation $\text{И}\overline{c}, c'.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge t'$, the permutation $\rho = (b\ c_1)$ is such that $\mathtt{dom}(\rho) \cap (\overline{c} \cup \{c'\}) = \emptyset$. Using Equivariance, we have $\text{И}\overline{c}, c'.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge (b\ c_1)\cdot t'$. By the definition of И, we can express this as $\text{И}c_1.(\text{И}\overline{c}, c'.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge (b\ c_1)\cdot t')$, implying $\text{И}\overline{c}, c', c_1.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge (b\ c_1)\cdot t'$. We can put $\text{И}c_1$ inside because c_1 is a fresh atom distinct from every atom occurring in $([\Delta]_{\curlywedge})_{A,\overline{c_0}}$, and thus, the result we aimed follows by rule ($\curlywedge$**abs**).

($\Leftarrow$) : If the last rule applied is ($\curlywedge$**abs**), then we have two cases.

- The first case is when $t \equiv [a]t'$. In this case, $\Delta \vdash a\#[a]t'$ follows trivially by rule ($\#[a]$).
- The other case is when $t \equiv [b]t'$. In this case, we have a derivation

$$\frac{\text{И}\overline{c}, c', c_1.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge (b\ c_1)\cdot t'}{\text{И}\overline{c}, c'.([\Delta]_{\curlywedge})_{A,\overline{c_0}} \vdash (a\ c') \curlywedge [b]t'}\ (\curlywedge\textbf{abs})$$

By the induction hypothesis, we have $\Delta \vdash a\#(b\ c_1)\cdot t'$ and we can build the derivation

$$\frac{\Delta \vdash a\#(b\ c_1)\cdot t'}{\Delta \vdash a\#[c_1](b\ c_1)\cdot t'}\ (\#abs)$$

Given the equivalence $\Delta \vdash a\#[c_1](b\ c_1)\cdot t' \iff \Delta \vdash a\#(b\ c_1)\cdot [b]t'$, we can apply the Equivariance of freshness with $\rho = (b\ c_1)$. This yields: $\Delta \vdash a\#[b]t'$. □

□

The next result states that derivation of certain fixed-point constraints can be mapped, via $[\cdot]_{\#}$, to derivations of freshness constraints, and vice-versa. Similarly for α-equality constraints.

Theorem 5.5 (From and to $[\cdot]_{\#}$) *The following hold, for $\overline{c_0} \subseteq \overline{c}$:*

(i) $И\overline{c}, c_1.\Upsilon_{A,\overline{c_0}} \vdash (a\ c_1) \curlywedge t \iff [\Upsilon_{A,\overline{c_0}}]_{\#} \vdash a\#t$.

(ii) $И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash s \overset{\curlywedge}{\approx} t \iff [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#\mathtt{Var}(s,t)} \vdash s \approx t$.

Proof. Again, in both cases the proof is by structural induction and follows by analysing the last rule applied in the derivation of the judgement. Here, we choose to prove only the interesting case involving the variable rule

(i) ($\Rightarrow$) : The last rule applied is ($\curlywedge$**var**):

$$\frac{\{\pi^{-1}(a)\} = \mathtt{dom}((a\ c')^{\pi'^{-1}}) \setminus \overline{c} \cup \{c'\} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))}{И\overline{c}, c'.\Upsilon_{A,\overline{c_0}} \vdash (a\ c') \curlywedge \pi' \cdot X}\ (\curlywedge\mathbf{var})$$

That is, by hypothesis, $Иc_{\pi'^{-1}(a)}.(\pi'^{-1}(a)\ c_{\pi'^{-1}(a)})\curlywedge \in \Upsilon_{A,\overline{c_0}}|_X$ and by the translation $[\cdot]_{\#}$ and $[\cdot]_{\curlywedge}$ we have $\pi'^{-1}(a)\#X \in [\Upsilon_{A,\overline{c_0}}|_X]_{\#} \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}$. Then $[\Upsilon_{A,\overline{c_0}}]_{\#} \vdash a\#\pi' \cdot X$ by rule ($\#var$).

($\Leftarrow$) : The last rule applied is ($\#var$):

$$\frac{\pi^{-1}(a)\#X \in [\Upsilon_{A,\overline{c_0}}]_{\#}}{[\Upsilon_{A,\overline{c_0}}]_{\#} \vdash a\#\pi \cdot X}\ (\#var)$$

By the translation, $Иc_{\pi'^{-1}(a)}.(\pi'^{-1}(a)\ \ c_{\pi'^{-1}(a)})\ \curlywedge\ X \in \Upsilon_{A,\overline{c_0}}|_X$ and so $\pi^{-1}(a)$ is in $\mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))$. Thus $\{\pi^{-1}(a)\} = \mathtt{dom}((a\ c')^{\pi^{-1}}) \setminus \{\overline{c}, c'\} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))$ and the result follows by rule ($\curlywedge$**var**).

(ii) ($\Rightarrow$) : The last rule applied is ($\overset{\curlywedge}{\approx}$ **var**):

In this case $t \equiv \pi \cdot X, s \equiv \pi' \cdot X, \mathtt{Var}(s,t) = \{X\}$ and there is a derivation

$$\frac{\mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))}{И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \cdot X \overset{\curlywedge}{\approx} \pi' \cdot X}\ (\overset{\curlywedge}{\approx}\mathbf{var})$$

We want to show that $[\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X} \vdash \pi \cdot X \approx \pi' \cdot X$. There are two cases to consider:

- If $\mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c} = \emptyset$, then $\mathtt{ds}(\pi, \pi') = \mathtt{dom}(\pi'^{-1} \circ \pi) \subseteq \overline{c}$. Consequently, $\mathtt{ds}(\pi, \pi')\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$ holds true because $\overline{c}\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$. Therefore, $[\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X} \vdash \pi \cdot X \approx \pi' \cdot X$ follows by ($\#var$).
- Now, suppose $\mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c} \neq \emptyset$, then there are atoms in $\mathtt{dom}(\pi'^{-1} \circ \pi)$ that are not in $\overline{c}$. If $b \in \mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c}$ then $b \in \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))$, so by the definition of $[\cdot]_{\#}$, we have $\mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c}\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}$. Consequently, $\mathtt{dom}(\pi'^{-1} \circ \pi) \setminus \overline{c}\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$. Additionally, $\overline{c}\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$. Hence, $\mathtt{dom}(\pi'^{-1} \circ \pi) \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$, and the result follows by ($\#var$) because $\mathtt{dom}(\pi'^{-1} \circ \pi) = \mathtt{ds}(\pi, \pi')$.

($\Leftarrow$) : The last rule applied is ($\#var$):

$$\frac{\mathtt{ds}(\pi, \pi')\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}}{[\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X} \vdash \pi \cdot X \approx \pi' \cdot X}\ (\#var)$$

We want to show that $И\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \pi \cdot X \overset{\curlywedge}{\approx} \pi' \cdot X$. In order to do this, we must show that $\mathtt{dom}(\pi^{-1} \circ \pi) \setminus \overline{c} \subseteq \mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))$.

From $\mathtt{ds}(\pi, \pi')\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}, \overline{\overline{c}\#X}$, we have $\mathtt{ds}(\pi, \pi') \setminus \overline{c}\#X \subseteq [\Upsilon_{A,\overline{c_0}}]_{\#}|_X$. By the translation $[\cdot]_{\#}$, we have that every atom in $\mathtt{ds}(\pi, \pi') \setminus \overline{c}$ is in $\mathtt{dom}(\mathtt{perm}(\Upsilon_{A,\overline{c_0}}|_X))$. Then the result follows because $\mathtt{ds}(\pi, \pi') = \mathtt{dom}(\pi'^{-1} \circ \pi)$. □

□

The following example shows that not all derivable strong fixed-point judgments can be mapped to a freshness judgement in the expected way.

Example 5.6 It is easy to see that the judgement $\mathsf{И}c_1, c_2.(a\ c_1) \curlywedge X, (b\ c_2) \curlywedge X \vdash (a\ b) \curlywedge X$ is derivable using the rule ($\curlywedge$**var**) from Figure 2 for $\mathtt{dom}((a\ b))\backslash\{c_1, c_2\} \subseteq \{a, b\}$. However, since neither a nor b are new names, we cannot use our translation to derive a freshness judgement as in Theorem 5.5(i). It follows from Theorem 5.3 (correctness) that there exists a derivation of the judgement $\mathsf{И}c_1, c_2.(a\ c_1) \curlywedge X, (b\ c_2) \curlywedge X \vdash (a\ b) \cdot X \overset{\curlywedge}{\approx} X$, and by Theorem 5.5(ii), there is a derivation of an equivalent α-equality judgement via #.

Since our rules in Figure 2 do not include a rule to deal with equational theories, the following result holds for the theory $\mathtt{T} = \mathsf{CORE}_{\curlywedge}$:

Theorem 5.7 *Nominal sets are sound denotational semantics for the nominal algebra with strong fixed-point constraints and judgements.*

Proof. The proofs follow from the translation $[\cdot]_{\curlywedge}$ and from the fact that nominal sets are sound denotational semantics for the nominal algebra with freshness constraints [16]. □

5.2 Using More Powerful Rules

In this section, we discuss an alternative approach to nominal algebra with fixed-point constraints that is also sound for all nominal set models. The changes proposed in Section 5.1 focused on restricting the form of the fixed-point contexts, which became equivalent to freshness contexts. Alternatively, we can keep the judgements and rules in Figure 1 as much as possible untouched and adapt only the problematic rules. Thus, in this subsection, we propose a change in the rule ($\curlywedge\mathsf{var}$) where we consider the permutation group generated by the permutations in the fixed-point context (denoted $\langle \mathtt{perm}(\Upsilon|_X)\rangle$) and the other rules are left as in Figure 1. The new rule is called $\textsc{Gvar}_{\curlywedge}$:

$$\frac{\pi^{\rho^{-1}} \in \langle \mathtt{perm}(\Upsilon|_X)\rangle}{\Upsilon \vdash \pi \curlywedge \rho \cdot X}\ \textsc{Gvar}_{\curlywedge}$$

In the proof of soundness (Theorem 4.1), we used the hypothesis of $\mathfrak{A}$ being a strong nominal algebra in the proof of the variable case. The next theorem shows that for the new set of rules, we do not need this hypothesis any longer.

Theorem 5.8 (Soundness) *Suppose* $\mathtt{T} = (\Sigma, Ax)$ *is a theory. Then*

(i) *If* $\Upsilon \vdash \pi \curlywedge t$ *then* $\Upsilon \vDash_{\mathtt{T}} \pi \curlywedge t$.

(ii) *If* $\Upsilon \vdash_{\mathtt{T}} t = u$ *then* $\Upsilon \vDash_{\mathtt{T}} t = u$.

Proof. In both cases, the proof is by induction on the rule applied. The interesting case is for the rule $\textsc{Gvar}_{\curlywedge}$**.** Then the derivation is

$$\frac{\pi^{\rho^{-1}} \in \langle \mathtt{perm}(\Upsilon|_X)\rangle}{\Upsilon \vdash \pi \curlywedge \rho \cdot X}\ \textsc{Gvar}_{\curlywedge}$$

Suppose that $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ is valid. We need to show that $\pi \cdot [\![\rho \cdot X]\!]^{\mathfrak{A}}_{\varsigma} = [\![\rho \cdot X]\!]^{\mathfrak{A}}_{\varsigma}$.

If $\mathtt{perm}(\Upsilon|_X) = \{\gamma_1, \ldots, \gamma_n\}$, then from the inclusion $\pi^{\rho^{-1}} \in \langle \mathtt{perm}(\Upsilon|_X)\rangle$, we get $\pi^{\rho^{-1}} = \gamma_1^{\alpha_1} \circ \ldots \circ \gamma_n^{\alpha_n}$, for $\alpha_1, \ldots, \alpha_n \in \mathbb{N}$. Moreover, the validity of $[\![\Upsilon]\!]^{\mathfrak{A}}_{\varsigma}$ implies that $\gamma_i \curlywedge_{\mathtt{sem}} [\![X]\!]^{\mathfrak{A}}_{\varsigma}$ for every $\gamma_i \curlywedge X \in \Upsilon$. Thus, $\pi^{\rho^{-1}} \cdot [\![X]\!]^{\mathfrak{A}}_{\varsigma} = [\![X]\!]^{\mathfrak{A}}_{\varsigma}$ follows from the fact that it is in the generated group. □ □

5.2.1 Expressive Power

The new $\textsc{Gvar}_{\curlywedge}$ rule is sound, but more restrictive. The next two examples (Example 5.9 and Example 5.10) are about judgements that are derivable using the rules in Figure 1, but not with the new version using $\textsc{Gvar}_{\curlywedge}$. These examples illustrate two different aspects where the group of generated permutations is more restrictive.

Example 5.9 It was possible to derive $(a\ b) \curlywedge X, (c\ d) \curlywedge X \vdash (a\ c) \curlywedge X$ with the rules in Figure 1 including ($\mathbf{var}_{\curlywedge}$), however, it is not possible to derive this judgement using the new $\textsc{Gvar}_{\curlywedge}$ rule because $(a\ c)$ is not in the group generated by the permutations $(a\ b)$ and $(c\ d)$, i.e., $(a\ c) \notin \langle (a\ b), (c\ d)\rangle = \{id, (a\ b), (c\ d), (a\ b) \circ (c\ d)\}$.

We recall the fact that the new names are not annotated with the quantifier $\mathsf{И}$ in Figure 1, the information about new names is relegated to the meta-language as usual when discussing languages with binders (such as the λ-calculus). As we can see below, this 'sugared' notation (without $\mathsf{И}$) for new names causes some problems:

Example 5.10 In this example we want to mimic the derivation in Figure 3 of $a, b\#X \implies (a\ b)\cdot[a]X \approx [a]X$. Suppose $\Upsilon = \{(b\ d') \curlywedge X\}$. It is possible to derive $(b\ d') \curlywedge X \vdash (a\ b)\cdot[a]X = [a]X$ using $\curlywedge$ via the rules of Figure 1. However, with the new rule $\textsc{Gvar}_{\curlywedge}$ the same judgement is not derivable: The first difficulty is that we can't inform Υ that b is fresh, the formal way would be to have the constraint $\mathsf{И}d'.(b\ d') \curlywedge X \in \Upsilon$. In the meta-language, this reduces to $(b\ d') \curlywedge X \in \Upsilon$ and records the information 'where d' is a new name'. However, this is not satisfactory since d' can be *any* new name, as we can see below.

$$\dfrac{\dfrac{(c_3\ c_1) \notin \langle (c_1\ c_2), (c_3\ c_4), (b\ d')\rangle}{\Upsilon, (c_1\ c_2) \curlywedge X, (c_3\ c_4) \curlywedge X \vdash (a\ c_1) \curlywedge (a\ c_3)\cdot X} \quad \dfrac{(b\ d_1) \notin \langle (d_1\ d_2), (c'_3\ c'_4), (b\ d')\rangle}{\Upsilon, (d_1\ d_2) \curlywedge X, (c'_3\ c'_4) \curlywedge X \vdash (b\ d_1) \curlywedge (a\ c'_3)\cdot X}}{\dfrac{\Upsilon, (c_1\ c_2) \curlywedge X \vdash (a\ c_1) \curlywedge [a]X \qquad \Upsilon, (d_1\ d_2) \curlywedge X \vdash (b\ d_1) \curlywedge [a]X}{\Upsilon \vdash (a\ b)\cdot[a]X = [a]X}\ (\mathbf{perm})}$$

where $c_1, c_2, c_3, c_4, d_1, d_2, c'_3, c'_4$ are new/fresh names.

Notice that none of the sub-derivation (in the branches) are possible. However, looking closely, the conditions needed to close the above derivation, i.e., $(c_3\ c_1) \in \langle (c_1\ c_2), (c_3\ c_4), (b\ d')\rangle$ and $(b\ d_1) \in \langle (d_1\ d_2), (c'_3\ c'_4), (b\ d')\rangle$, aim to check whether b and c_3 are fresh for X, and this information could be derived from the context, however, using different new names ($(b\ d')$ is in Υ instead of $(b\ d_1)$). The problem is that the new names do not interact with the old names when generating the group of permutations.

We conjecture that a simple solution for both examples would consist of a combination of strong fixed-point contexts together with the generation of a larger group of permutations: including the group of permutations generated by the new atoms of Example 5.10, i.e., to consider the permutation group $\langle \mathtt{Perm}(\{c_1, c_2, c_3, c_4, c'_3, c'_4, d_1, d_2, d'\}), (b\ d')\rangle$. Similarly, for Example 5.9 we would need to specify the new atoms in the context and generate the permutation group of these new atoms. However, we need to check if relaxing the group generator restriction in this way does not introduce inconsistencies.

6 Discussion, Applications and Future Work

In the standard nominal approach, using freshness constraints, nominal unification problems [22], denoted Pr, are finite sets of constraints of the form $s \approx^? t$ and $a\#^? t$. Solutions to (nominal) unification problems are pairs (Δ, σ) consisting of a freshness context Δ and a substitution σ such that: (i) $\Delta \vdash s\sigma \approx t\sigma$, for each $s \approx^? t \in \mathsf{Pr}$; (ii) $\Delta \vdash a\#t\sigma$, for each $s\#^? t \in \mathsf{Pr}$; and (iii) $\Delta \vdash X\sigma \approx X\sigma\sigma$, for all $X \in \mathtt{Var}(\mathsf{Pr})$. Derivability in items (i)-(iii) is established by the rules in Figure 3.

We are interested in nominal equational unification problems, that is, in solving equations in the ground term algebra when in addition to α-equivalence and freshness constraints, one is also interested in equational theories (such as commutativity, associativity, etc). Nominal equational unification problems, denoted $\mathsf{Pr}_{\mathtt{E}}$, are finite sets of constraints $s \approx^?_{\mathtt{E}} t$ and $a\#^? t$. Solutions of $\mathsf{Pr}_{\mathtt{E}}$ are defined in terms of $\approx_{\mathtt{E}}$, as expected, and rely on extending the rules in Figure 3 with the appropriate rules for the equational theory

$\mathtt{E}$. For example, in the case of the commutative theory $\mathtt{C}$: first we need to specify the commutative function symbol $\mathtt{f}^{\mathtt{C}}$ and the axiom that establishes the commutativity for this symbol $\mathtt{C} = \{ \vdash \mathtt{f}^{\mathtt{C}}(X,Y) = \mathtt{f}^{\mathtt{C}}(Y,X)\}$. Now we extend and modify the rules of Figure 3 as follows: (i) replace $\approx$ for $\approx_{\mathtt{C}}$ in all the rules; (ii) add the rule $\dfrac{\Delta \vdash t_1 \approx_{\mathtt{C}} s_i \qquad \Delta \vdash t_2 \approx_{\mathtt{C}} s_{3-i}}{\Delta \vdash \mathtt{f}^{\mathtt{C}}(t_1,t_2) \approx_{\mathtt{C}} \mathtt{f}^{\mathtt{C}}(s_1,s_2)}$. Together, these rules implement $\alpha, \mathtt{C}$-equality.

Nominal unification algorithms are defined in terms of simplification rules that can be seen as bottom-up applications of derivation rules defining $\approx$ and $\#$. More details can be found in [3].

Example 6.1 Consider the nominal $\mathtt{C}$-unification problem $\mathsf{Pr}_{\mathtt{C}} = \{\mathtt{f}^{\mathtt{C}}(X,Y) \approx^?_{\mathtt{C}} \mathtt{f}^{\mathtt{C}}(c,(a\ b)\cdot X)\}$. This problem reduces to two simpler problems:

$$\mathsf{Pr}_{\mathtt{C}} \Longrightarrow \begin{cases} \{X \approx^?_{\mathtt{C}} c, Y \approx^?_{\mathtt{C}} (a\ b)\cdot X\} \\ \{Y \approx^?_{\mathtt{C}} c, X \approx^?_{\mathtt{C}} (a\ b)\cdot X\} \end{cases}$$

The top branch has a solution $\{X \mapsto c, Y \mapsto c\}$. The bottom branch is more problematic due to the fixed-point equation $\{X \approx^?_{\mathtt{C}} (a\ b)\cdot X\}$.

We have seen in previous works on nominal unification (without equational theories) that the usual simplification rule for fixed-point equations on variables is $\pi \cdot X \approx \pi' \cdot X \Longrightarrow ds(\pi,\pi')\#X$. However, this rule is not complete for the commutative case, since we lose solutions such as $X \mapsto \mathtt{f}^{\mathtt{C}}(a,b)$, $X \mapsto \mathtt{f}^{\mathtt{C}}(\mathtt{f}^{\mathtt{C}}(a,b),\mathtt{f}^{\mathtt{C}}(a,b))$, $X \mapsto \forall[c]g(\mathtt{f}^{\mathtt{C}}(a,b),c)$, etc. We have shown in [2] that there are infinite independent solutions for this problem.

Currently, there are two approaches to solving nominal equational problems:

(i) Do not attempt to solve fixed-point equations and leave them as part of solutions [3]. Thus, solutions of unification problems would be triples (Δ,σ,P), where P is a finite set of fixed-point equations of the form $X \approx_E \pi\cdot X$, and we can use extension of Figure 3 to validate solutions.

(ii) Use a different and more expressive formalism, using fixed-point constraints that capture fixed-point equations [4]. Solutions to unification problems are again pairs (Υ,σ), but with fixed-point contexts, and solutions are validated using the rules in Figure 1 taking into account the axioms defining the theory $\mathtt{E}$.

First, with the approach (i) the solutions to Example 6.1 are expressed by the triples $(\emptyset,\{X\mapsto c, Y\mapsto c\},\emptyset)$ and $(\emptyset,\{Y\mapsto c\},\{X\approx^?_{\mathtt{C}}(a\ b)\cdot X\})$. Second, with the approach (ii) the solutions to the problem in Example 6.1 are expressed by the pairs $(\emptyset,\{X\mapsto c, Y\mapsto c\})$ and $(\{(a\ b)\curlywedge X\},\{Y\mapsto c\}\})$, and concrete instances δ of X that validate $(a\ b)\curlywedge X$, when composed with $\{Y\mapsto c\}$ (i.e. $\{Y\mapsto c\}\delta$), are also solutions to the initial problem.

6.1 A sound fixed-point approach for commutativity: group generator

By Theorem 4.1, fixed-point constraints are a sound approach for proving validity of judgements in strong models, and these do not accept permutative theories, such as commutativity. But we can reason modulo the theories defined by the axioms in Example 4.3.

The use of more powerful rules proposed in Section 5.2 might be useful for commutativity, although limited in terms of derivable judgements. First we consider the following modification of the rules of Figure 1:

- Consider the **(ax)** rule with the axiom $\mathtt{C} = \{\mathtt{f}^{\mathtt{C}}(X,Y) = \mathtt{f}^{\mathtt{C}}(Y,X)\}$, where $\mathtt{f}^{\mathtt{C}} \in \Sigma$;
- Replace the **($\curlywedge$var)** rule for $\textsc{Gvar}_{\curlywedge}$.

The soundness is trivial, as it is a direct consequence of Theorem 5.8.

6.2 Another sound fixed-point approach for commutativity: strong judgements

The use of strong judgements as proposed in Section 5.1 also seems promising for dealing with commutativity. First we consider the following modification of the rules of Figure 2:

- Replace $\overset{\curlywedge}{\approx}$ for $\overset{\curlywedge}{\approx}_{\mathtt{C}}$ in all the rules.
- Replace $\curlywedge$ for $\curlywedge_{\mathtt{C}}$ in all the rules.
- Add the following rule for commutativity $\dfrac{\mathit{И}\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash t_1 \overset{\curlywedge}{\approx}_{\mathtt{C}} s_i \qquad \mathit{И}\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash t_2 \overset{\curlywedge}{\approx}_{\mathtt{C}} s_{3-i}}{\mathit{И}\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash \mathtt{f}^{\mathtt{C}}(t_1,t_2) \overset{\curlywedge}{\approx}_{\mathtt{C}} \mathtt{f}^{\mathtt{C}}(s_1,s_2)}$

We conjecture that we can extend the translations defined in Section 5.1 to take into account the commutativity theory in order to obtain soundness by extending Theorem 5.7. The expected extension of the translation would map judgements of the form $\mathit{И}\overline{c}.\Upsilon_{A,\overline{c_0}} \vdash (a\ b) \curlywedge_{\mathtt{C}} X$ where b is not $\mathit{И}$-quantified (and that are not equivalent to a freshness judgments) to a fixed-point equation $(a\ b)\cdot X \overset{\curlywedge}{\approx}_{\alpha,\mathtt{C}} X$. (cf. Example 5.6). This provides a foundation for unification algorithms modulo α and $\mathtt{C}$.

7 Conclusion

We formalised the intentional semantics for fixed-point constraints and verified that strong nominal sets are a sound denotational semantics for nominal algebra with fixed-point constraints. We proposed alternatives to regain soundness for the whole class of nominal sets, by proposing more powerful rules or stronger judgements and new rules. Our developments provide a sound approach for solving nominal equational problems modulo strong theories. However, the best approach to deal with non-strong theories, such as the permutative theories, still have to be investigated. Investigations towards completeness are under development and will be reported in future work.

References

[1] Andrew M. Pitts, *"Nominal Sets: Names and Symmetry in Computer Science"*, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press (2013).
https://doi.org/10.1017/CBO9781139084673

[2] Ayala-Rincón, M., W. de Carvalho Segundo, M. Fernández and D. Nantes-Sobrinho, *On solving nominal fixpoint equations*, in: C. Dixon and M. Finger, editors, *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*, pages 209–226, Springer (2017).
https://doi.org/10.1007/978-3-319-66167-4_12

[3] Ayala-Rincón, M., W. de Carvalho Segundo, M. Fernández, G. F. Silva and D. Nantes-Sobrinho, *Formalising nominal C-unification generalised with protected variables*, Math. Struct. Comput. Sci. **31**, pages 286–311 (2021).
https://doi.org/10.1017/S0960129521000050

[4] Ayala-Rincón, M., M. Fernández and D. Nantes-Sobrinho, *On nominal syntax and permutation fixed points*, Log. Methods Comput. Sci. **16**, page 19:1–19:36 (2020).
https://doi.org/10.23638/LMCS-16(1:19)2020

[5] Ayala-Rincón, M., M. Fernández, D. Nantes-Sobrinho and D. Vale, *On solving nominal disunification constraints*, in: A. P. Felty and J. Marcos, editors, *Proceedings of the 14th Workshop on Logical and Semantic Frameworks with Applications, LSFA 2019, Natal, Brazil, August, 2019*, volume 348 of *Electronic Notes in Theoretical Computer Science*, pages 3–22, Elsevier (2019).
https://doi.org/10.1016/j.entcs.2020.02.002

[6] Ayala-Rincón, M., M. Fernández, D. Nantes-Sobrinho and D. Vale, *Nominal equational problems*, in: S. Kiefer and C. Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 22–41, Springer (2021).
https://doi.org/10.1007/978-3-030-71995-1_2

[7] Ayala-Rincón, M., W. de Carvalho-Segundo, M. Fernández, D. Nantes-Sobrinho and A. C. Rocha-Oliveira, *A formalisation of nominal alpha-equivalence with A, C, and AC function symbols*, Theoretical Computer Science **781**, pages 3–23 (2019), ISSN 0304-3975. Logical and Semantic Frameworks with Applications.
https://doi.org/https://doi.org/10.1016/j.tcs.2019.02.020

[8] Calvès, C. and M. Fernández, *Matching and alpha-equivalence check for nominal terms*, J. Comput. Syst. Sci. **76**, pages 283–301 (2010).
https://doi.org/10.1016/J.JCSS.2009.10.003

[9] Cheney, J. and C. Urban, *alpha-prolog: A logic programming language with names, binding and a-equivalence*, in: B. Demoen and V. Lifschitz, editors, *Logic Programming, 20th International Conference, ICLP 2004, Saint-Malo, France, September 6-10, 2004, Proceedings*, volume 3132 of *Lecture Notes in Computer Science*, pages 269–283, Springer (2004).
https://doi.org/10.1007/978-3-540-27775-0_19

[10] Comon, H., *Disunification: A survey*, in: J. Lassez and G. D. Plotkin, editors, *Computational Logic - Essays in Honor of Alan Robinson*, pages 322–359, The MIT Press (1991).
https://api.semanticscholar.org/CorpusID:118923989

[11] Comon, H. and P. Lescanne, *Equational problems and disunification*, J. Symb. Comput. **7**, pages 371–425 (1989).
https://doi.org/10.1016/S0747-7171(89)80017-3

[12] Fernández, M., *Narrowing based procedures for equational disunification*, Appl. Algebra Eng. Commun. Comput. **3**, pages 1–26 (1992).
https://doi.org/10.1007/BF01189020

[13] Fernández, M., *AC complement problems: Satisfiability and negation elimination*, Journal of Symbolic Computation **22**, pages 49–82 (1996), ISSN 0747-7171.
https://doi.org/https://doi.org/10.1006/jsco.1996.0041

[14] Fernández, M. and M. J. Gabbay, *Nominal rewriting*, Inf. Comput. **205**, pages 917–965 (2007).
https://doi.org/10.1016/j.ic.2006.12.002

[15] Gabbay, M. J., *Nominal algebra and the HSP theorem*, J. Log. Comput. **19**, pages 341–367 (2009).
https://doi.org/10.1093/LOGCOM/EXN055

[16] Gabbay, M. J. and A. Mathijssen, *Nominal (universal) algebra: Equational logic with names and binding*, J. Log. Comput. **19**, pages 1455–1508 (2009).
https://doi.org/10.1093/logcom/exp033

[17] Gabbay, M. J. and A. M. Pitts, *A new approach to abstract syntax with variable binding*, Formal Aspects Comput. **13**, pages 341–363 (2002).
https://doi.org/10.1007/s001650200016

[18] Kirchner, C., R. Kopetz and P.-E. Moreau, *Anti-pattern matching modulo*, in: C. Martín-Vide, F. Otto and H. Fernau, editors, *Language and Automata Theory and Applications*, pages 275–286, Springer Berlin Heidelberg, Berlin, Heidelberg (2008), ISBN 978-3-540-88282-4.
https://doi.org/10.1007/978-3-540-88282-4_26

[19] Levy, J. and M. Villaret, *An efficient nominal unification algorithm*, in: C. Lynch, editor, *Proceedings of the 21st International Conference on Rewriting Techniques and Applications, RTA 2010, July 11-13, 2010, Edinburgh, Scottland, UK*, volume 6 of *LIPIcs*, pages 209–226, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2010).
https://doi.org/10.4230/LIPIcs.RTA.2010.209

[20] Pitts, A. M., *Nominal logic, a first order theory of names and binding*, Inf. Comput. **186**, pages 165–193 (2003).
https://doi.org/10.1016/S0890-5401(03)00138-X

[21] Shinwell, M. R., A. M. Pitts and M. Gabbay, *Freshml: programming with binders made simple*, in: C. Runciman and O. Shivers, editors, *Proceedings of the Eighth ACM SIGPLAN International Conference on Functional Programming, ICFP 2003, Uppsala, Sweden, August 25-29, 2003*, pages 263–274, ACM (2003).
https://doi.org/10.1145/944705.944729

[22] Urban, C., A. M. Pitts and M. Gabbay, *Nominal unification*, Theor. Comput. Sci. **323**, pages 473–497 (2004).
https://doi.org/10.1016/j.tcs.2004.06.016