Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 348 (2020) 3–22

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# On Solving Nominal Disunification Constraints [1]

Mauricio Ayala-Rincón[†,‡], Maribel Fernández[*], Daniele Nantes-Sobrinho[†] and Deivid Vale[†2]

[†] *Departamentos de Matemática e* [‡] *Ciência da Computação*
*Universidade de Brasília, Brasília-DF, Brazil*
[*] *Department of Informatics, King's College London, London, UK*

**Abstract**

This paper proposes an extension of first-order disunification problems by taking into account binding operators according to the nominal approach. In this approach, bindings are implemented through atom abstraction, and renaming of atoms is implemented via atom permutations. In the nominal setting, unification problems consist of equational questions ($s \approx_\alpha^? t$) considered under freshness constraints ($a\#^? t$) that restrict solutions by forbidding free occurrences of atoms in the instantiation of variables. In addition to equational and freshness constraints, nominal disunification problems include also nominal disunification constraints ($s \not\approx_\alpha^? t$), and their solutions consist of a substitution and additional freshness constraints such that under these constraints the instantiation of the equations, disequations and freshness constraints with the substitution hold. By re-using nominal unification techniques, this paper shows how to decide whether two nominal terms can be made different modulo $\alpha$-equivalence. This is done by extending previous results on first-order disunification, and defining the notion of *solutions with exceptions* in the nominal syntax. A discussion on the semantics of disunification constraints is also given.

*Keywords:* Nominal Logic, Nominal Semantics, Unification, Disunification

## 1 Introduction

Nominal techniques can be used to reason about systems with binders. The binding structure of these systems always requires a method to deal with $\alpha$-equivalence between objects in the system, i.e., objects (usually parse tree representation of the concrete syntax) are considered equal if they differ only by the name of bound variables. For instance, in the syntax of $\lambda$-calculus, terms like $\lambda x.x\,y$ and $\lambda z.z\,y$ should be considered equivalent, despite their syntactical differences.

It is common in the literature to consider the $\alpha$-equivalence relation as part of the syntactical structure of terms of the language. One often says "terms are

[1] Work partially funded by CNPq

[2] Email: deividvale@mat.unb.br, {dnantes,ayala}@unb.br, maribel.fernandez@kcl.ac.uk

https://doi.org/10.1016/j.entcs.2020.02.002

considered syntactically identical if they are $\alpha$-convertible". This means that one considers the quotient of the set of terms by the $\alpha$-equivalence relation. So one has the problem of which representative of $\alpha$-equivalence classes should be chosen. One of the most popular strategies to solve this problem is called the "Barendregt Variable Convention": choose representatives for which the bound variables are mutually distinct and distinct from any *free* variable in the current context. This strategy solves the problem for handwritten proofs and calculations, but not for implementations. Another treatment of $\alpha$-equivalence is to get rid of equivalence classes by considering *de Bruijn indices* instead of variable names. Using de Bruijn indices, free and bound variables are indexed as naturals and thus all objects have unique representations so that one does not need to worry about representatives of equivalence classes. This approach facilitates the implementation of systems with binders but at the cost of readability.

The nominal approach diverges from those above in two important ways: first, one can reason about $\alpha$-equivalence in a readable way, very close to informal practice, while still remaining fully formal since $\alpha$-renaming is embedded in the nominal syntax; second, nominal $\alpha$-equivalence is easy to implement in computer systems.

Nominal terms have *atoms* $(a, b, c \ldots)$ used to represent object-level variables, and *variables, or unknowns* $(X, Y, \ldots)$ used to express variables on the meta-level. Atoms can be abstracted by a binder operator but cannot be instantiated by a substitution, whereas variables cannot be abstracted but can be instantiated by a substitution. For instance, the nominal term $[a]\, t$ represents the abstraction of atom $a$ in $t$. To rename an atom $a$ to another atom $b$ we make use of an *atom permutation*. Permutations are built as lists of atom *swappings* of the form $(a\ b)$. The action of $\pi = (a\ b)$ over $[a]\, t$, denoted by $\pi \cdot t$ gives as result the term $[b]\, t'$ where $t'$ is obtained by the replacement of all occurrences of $a$ by $b$ and all occurrences of $b$ by $a$ in $t$. The action of an atom permutation $\pi$ over a meta-variable $X$ will be 'suspended' in $X$, written as $\pi \cdot X$, and will be 'executed' only when $X$ is instantiated. The $\alpha$-equivalence relation over nominal terms is built using permutations and a *freshness relation* between atoms and terms, written as $a\#t$, which means that $a$ cannot occur *free* in $t$.

Nominal techniques have been widely explored and investigated for the last years [12,3,17]. Nominal unification has also been developed [19,8], and more recently, works on unification modulo equational theories [1,2,4] have also been developed. Unification is the problem of finding a substitution $\sigma$ that makes two terms 'equal' i.e., $s\sigma \approx_\alpha t\sigma$. In the nominal setting, $\alpha$-equality comes with freshness conditions for atoms, which should be taken into account when dealing with nominal unification problems. For instance, the problem of unifying $\lambda\,[a]\, X$ and $\lambda\,[b]\, Y$ reduces to the problem of unifying $X \approx_\alpha{}^{?}\ (b\ a) \cdot Y$ under the condition that $a\#Y$. Therefore, a solution to a nominal unification problem will be a pair $\langle \Gamma, \sigma \rangle$ consisting of a set of freshness constraints $\Gamma$, and a substitution over variables $\sigma$. Several applications of nominal unification exist, for instance, in logic programming, automatic deduction, theorem proving, among others.

This work is about *nominal disunification*, that is, the problem of solving nominal

unification questions enriched with *disequations*, i.e., constraints of the form $s \not\approx_\alpha^? t$. For example, consider the unification problem $\lambda\,[a]\,X \approx_\alpha^? \lambda\,[b]\,Y$ as above, but imposing the condition that solutions can neither map $X$ to the atom $a$ nor $Y$ to the atom $b$. This condition may be given as a set of disequations, and solutions should be computed in such a way that they (and therefore, their instances) satisfy the imposed restriction. The *nominal disunification constraint* is then represented as:

$$\left\langle \lambda\,[a]\,X \approx_\alpha^? \lambda\,[b]\,Y \;\;||\;\; X \not\approx_\alpha^? a,\;\; Y \not\approx_\alpha^? b \right\rangle$$

Imposing such conditions has some side-effects that need to be addressed in order to be able to formally state a definition of solution. In a more general view, a nominal disunification problem is given as $\mathcal{P} = \langle \Delta \vdash s_1 \approx_\alpha^? t_1, \ldots, s_n \approx_\alpha^? t_n \mid \nabla \vdash u_1 \not\approx_\alpha^? v_1, \ldots, u_m \not\approx_\alpha^? v_m \rangle$, and a solution to such problem is a pair $\langle \Gamma, \sigma \rangle$ of a context $\Gamma$ and a substitution $\sigma$, such that $\sigma$ makes terms of each equation equal, but leaves those of the disequations different, while satisfying the freshness constraints $\Delta$ and $\nabla$.

The strategy proposed by Buntine and Bürckert [6] to solve systems containing first-order equations and disequations is followed in the current work. But its extension to the nominal setting is not straightforward since the notions of equality and disequality are different and the freshness side conditions add extra constraints to the problem. The standard nominal unification algorithm [19] can be reused to provide solutions to nominal unification problems, and following Buntine and Bürckert's approach, we show that nominal disequations can be treated in a nominal term-algebra.

The main contributions of this paper can be summarized as follows.

(i) We extend first-order disunification problems to the nominal framework introducing *nominal constraint problems*.

(ii) From the semantics point of view, we show that Birkhoff's HSP Theorem (Theorem 2.14), as in the first-order case, does not hold for nominal disequations (Example 2.16).

(iii) We extend the notion of substitution with exceptions to *solution pairs* that consist of a freshness context and a substitution, with exceptions (Definition 3.5). In addition, a version of the Consistency Test Algorithm (Algorithm 1) to deal with pairs with exceptions is proposed.

(iv) We propose a sound, complete, and terminating (provided nominal unification is finitary) procedure (Algorithm 2) to solve nominal disunification constraints that reuses the nominal unification algorithm.

(v) We prove that the Representation Theorem holds in the nominal approach to disunification (Theorem 4.3).

**Outline of the paper**

Section 2 establishes the main required notions on nominal syntax and semantics, equality and unification. Section 3 introduces the nominal constraint problems as well

as a generalized notion of instantiation and proves some results on the consistency of pairs with exceptions. Section 4 shows how to solve nominal constraint problems by reusing the nominal unification algorithm. Section 5 describes related work and Section 6 concludes the paper.

## 2 Preliminaries

We assume the reader is familiar with nominal syntax and sets and recall the main concepts and notations that are needed in the paper; for more details we refer the reader to [12,17].

### *2.1 Nominal Syntax*

Fix countable disjoint sets of *variables* $\mathbb{X} = \{X, Y, Z, \dots\}$ and *atoms* $\mathbb{A} = \{a, b, c, d, \cdots\}$. Variables represent meta-level *unknowns* and atoms object level variable symbols. Atoms are identified by their name, so it will be redundant to say two atoms $a$ and $b$ are different. A signature $\Sigma$ is a set of term-formers such that each $f \in \Sigma$ is assigned a unique non-negative integer $n$, called the *arity* of $f$, written as $f : n$.

A permutation $\pi$ is a bijection $\mathbb{A} \to \mathbb{A}$ with finite domain, i.e., the set $\mathtt{supp}(\pi) := \{a \in \mathbb{A} \mid \pi(a) \neq a\}$ is finite. Write $\mathtt{id}$ for the identity permutation. The composition of two permutations $\pi$ and $\pi'$ will be denoted as $\pi \circ \pi'$.

**Definition 2.1** [Nominal Terms] Let $\Sigma$ be a signature disjoint from $\mathbb{A}$ and $\mathbb{X}$. The set $T(\Sigma, \mathbb{A}, \mathbb{X})$ of nominal terms is inductively generated by the following grammar:

$$s, t, u, v \ ::= \ a \mid \pi \cdot X \mid [a]\, t \mid f(t_1, \dots, t_n)$$

where $a$ is an atom term, $\pi \cdot X$ is a moderated variable (or suspension), $[a]\, t$ denotes the abstraction of the atom $a$ in the term $t$, and $f(t_1, \dots, t_n)$, for $f : n$ in $\Sigma$, is a function application.

**Example 2.2** Let $\Sigma_\lambda := \{\mathtt{lam} : 1, \mathtt{app} : 2\}$ be the signature for the $\lambda$-calculus (for a complete axiomatization of the $\lambda$-calculus within the nominal syntax the reader is referred to [14]). If one consider $\lambda$-variables as atoms $\lambda$-terms can be inductively generated by the grammar:

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

To simplify notation, write $\mathtt{app}(s, t)$ as $s\, t$ and $\mathtt{lam}([a]\, s)$ as $\lambda\, [a]\, s$. The following are examples of nominal terms:

$$(\lambda\, [a]\, a)\, X \qquad (\lambda\, [a]\, (\lambda\, [b]\, b\, a)\ c)\, d$$

**Definition 2.3** [Action of permutations] The **object-level action** of a permutation

$\pi$ on a term $t$ is defined by induction on the structure of $t$, as follows:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\gamma \cdot X) \equiv (\pi \circ \gamma) \cdot X \qquad \pi \cdot [a]\, t \equiv [\pi \cdot a]\, (\pi \cdot t)$$
$$\pi \cdot f(t_1, \ldots, t_n) \equiv f(\pi \cdot t_1, \ldots, \pi \cdot t_n)$$

As usual, substitutions are defined as maps from variables to nominal terms with finite domain. We inductively extend the action of a substitution from variables to terms, in the following way:

**Definition 2.4** [Action of substitutions] The action of a substitution $\sigma$ on a term $t$, denoted as $t\sigma$, is inductively defined by:

$$a\sigma \equiv a \quad (\pi \cdot X)\sigma \equiv \pi \cdot (X\sigma) \quad ([a]\, t)\sigma \equiv [a]\, (t\sigma) \quad f(t_1, \ldots, t_n)\sigma \equiv f(t_1\sigma, \ldots, t_n\sigma)$$

Substitutions are written in postfix notation: $t(\sigma\gamma) \equiv (t\sigma)\gamma$.

### *2.2 Equality and Derivability*

Equality in nominal terms corresponds in fact to $\alpha$-equality and relies on the notion of "fresh variable". Two fundamental predicates are *freshness*, $\#$, and *alpha-equality*, $\approx_\alpha$:

- $a\#t$ means that the atom $a$ cannot occur free in $t$;
- $s \approx_\alpha t$ means that $s$ is $\alpha$-equivalent to $t$.

*Constraints* are generated by the grammar

$$P, Q, C := a\#t \mid s \approx_\alpha t$$

The first constraint $a\#t$ is called a *freshness constraint* whereas $s \approx_\alpha t$ is called an *$\alpha$-equality constraint*. A freshness constraint of the form $a\#X$ or $a\#a$ is called *primitive*. Validity of freshness and equality constraints is specified via the natural deduction rules in Figures 1 and 2.

$$\frac{}{a\#b}\,(\#ab) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi \cdot X}\,(\#X) \qquad \frac{}{a\#\,[a]\, t}\,(\#a) \qquad \frac{a\#t}{a\#\,[b]\, t}\,(\#b)$$
$$\frac{a\#t_1 \cdots a\#t_n}{a\#f(t_1, \ldots, t_n)}\,(\#f)$$

Fig. 1. Derivation rules for freshness

To define $\approx_\alpha$ we use the *difference set* of two permutations

$$\mathtt{ds}(\pi, \gamma) := \{a \in \mathbb{A} \mid \pi(a) \neq \gamma(a)\}.$$

In the rules defining $\approx_\alpha$ below, $\mathtt{ds}(\pi, \gamma)\#X$ denotes the set of constraints $\{a\#X \mid a \in \mathtt{ds}(\pi, \gamma)\}$.

A finite set of primitive freshness constraints is called a *freshness context*. Write $\Delta, \Gamma$ and $\nabla$ for freshness contexts. We say a context $\Delta$ is *consistent* if it has no occurrence of $a\#a$, for any atom $a$.

$$\frac{}{a \approx_\alpha a}\ (\#ab) \qquad \frac{\mathtt{ds}(\pi,\gamma)\#X}{\pi \cdot X \approx_\alpha \gamma \cdot X}\ (\mathrm{Ds})$$

$$\frac{t_1 \approx_\alpha u_1 \cdots t_n \approx_\alpha u_n}{f(t_1,\ldots,t_n) \approx_\alpha f(u_1,\ldots,u_n)}\ (\mathrm{F})$$

$$\frac{t \approx_\alpha u}{[a]\,t \approx_\alpha [a]\,u}\ (\text{Abs-a}) \qquad \frac{(b\ a)\cdot t \approx_\alpha u \qquad b\#t}{[a]\,t \approx_\alpha [b]\,u}\ (\text{Abs-b})$$

Fig. 2. Derivation rules for $\alpha$-equivalence

**Definition 2.5** [Nominal judgment] Nominal algebra has two judgment forms:

(i) A freshness judgment form $\Delta \vdash a\#t$ is a pair with a context $\Delta$ and a freshness constraint $a\#t$.

(ii) An $\alpha$-equality judgment form $\Delta \vdash s \approx_\alpha t$ is a pair of a context $\Delta$ and an $\alpha$-equality constraint $s \approx_\alpha t$.

The judgment $\Delta \vdash a\#t$ (resp. $\Delta \vdash s \approx_\alpha t$) means that there exists a proof for $a\#t$ (resp. $a \approx_\alpha t$) using the rules of Figure 1 (resp. Figure 2) and the constraints of $\Delta$ as assumptions. We say that $\Delta$ *entails* $a\#t$ (resp. $s \approx_\alpha t$) or that the constraint is *derivable* from $\Delta$. As usual, $\Delta \nvdash C$ denotes that the constraint $C$ cannot be derived from $\Delta$. The notion of derivability can be extended to sets of constraints as expected.

### *2.3 Simplification rules for nominal unification*

Nominal unification is the problem of making two nominal terms, say $s$ and $t$, $\alpha$-equivalent, denoted as $s \approx_\alpha{}^? t$. Deciding this may require to decide whether an atom $a$ is fresh for a term $t$, denoted as $a\#^? t$. Since $t$ may contain variables, a freshness context $\Delta$ is necessary. The content of this section is taken from [12].

**Definition 2.6** [Nominal Unification Problem] A *nominal unification problem* $Pr$ is a pair $\langle \Delta, P\rangle$ consisting of a freshness context $\Delta$ and a finite set $P$ of freshness and $\alpha$-equality constraints of the form $a\#^? t$ and $s \approx_\alpha{}^? t$, respectively.

A sound and complete procedure for nominal unification was proposed in [19] and consists of applying the Simplification Rules of Figure 3. For freshness contexts $\nabla$ and $\nabla'$ and a substitution $\sigma$, the instance $\nabla\sigma$ denotes the set $\{a\#X\sigma \mid a\#X \in \nabla\}$ and $\nabla \vdash \nabla'\sigma$ denotes $\nabla \vdash a\#X\sigma$, for all $a\#X \in \nabla'$.

**Definition 2.7** A *solution* for a nominal unification problem $Pr = \langle \nabla, P\rangle$ is a pair of the form $\langle \Gamma, \sigma\rangle$ where $\Gamma$ is a consistent context and $\sigma$ a substitution such that the following conditions hold:

(i) $\Gamma \vdash \nabla\sigma$;

(ii) $\Gamma \vdash a\#t\sigma$, for all $a\#^? t \in Pr$;

(iii) $\Gamma \vdash t\sigma \approx_\alpha s\sigma$, for all $s \approx_\alpha{}^? t \in Pr$;

(iv) $X\sigma \equiv X\sigma\sigma$.

If there is no such $\langle \Gamma, \sigma \rangle$ we say that $Pr$ is *unsolvable*.

Let $Pr$ be a unification problem as above. We write $\mathcal{U}(Pr)$ to denote the set of all solutions of $Pr$. Solutions in $\mathcal{U}(Pr)$ are compared by the following partial order, called *instantiation ordering*.

**Definition 2.8** Let $\Gamma_1, \Gamma_2$ be consistent contexts, and $\sigma_1, \sigma_2$ substitutions. Then $\langle \Gamma_1, \sigma_1 \rangle \leq \langle \Gamma_2, \sigma_2 \rangle$ when there exists a substitution $\delta$ such that

$$\text{for all } X \in \mathbb{X},\ \Gamma_2 \vdash X\sigma_1\delta \approx_\alpha X\sigma_2 \qquad \text{and} \qquad \Gamma_2 \vdash \Gamma_1\delta$$

In this case we say $\langle \Gamma_2, \sigma_2 \rangle$ is *an instance* of the pair $\langle \Gamma_1, \sigma_1 \rangle$ on $\mathbb{X}$.

**Definition 2.9** A principal (or most general) solution to a problem $Pr$ is a least element of $\mathcal{U}(Pr)$.

$$\begin{aligned}
a\#^? b, Pr &\Longrightarrow Pr \\
a\#^? \pi \cdot X, Pr &\Longrightarrow \pi^{-1}(a)\#^? X, Pr \quad \pi \neq \mathtt{id} \\
a\#^? [a]\, t, Pr &\Longrightarrow Pr \\
a\#^? [b]\, t, Pr &\Longrightarrow a\#^? t, Pr \\
a\#^? f(t_1, \cdots, f_n), Pr &\Longrightarrow a\#^? t_1, \cdots, a\#^? t_n, Pr \\
a \approx_\alpha{}^? a, Pr &\Longrightarrow Pr \\
\pi \cdot X \approx_\alpha{}^? \gamma \cdot X, Pr &\Longrightarrow \mathtt{ds}(\pi, \gamma)\#X, Pr \\
f(s_1, \cdots, s_n) \approx_\alpha{}^? f(t_1, \cdots, t_n), Pr &\Longrightarrow s_1 \approx_\alpha{}^? t_1, \cdots, s_n \approx_\alpha{}^? t_n, Pr \\
[a]\, t \approx_\alpha{}^? [a]\, u, Pr &\Longrightarrow t \approx_\alpha{}^? u, Pr \\
[b]\, l \approx_\alpha{}^? [a]\, r, Pr &\Longrightarrow (a\ b) \cdot l \approx_\alpha{}^? r, a\#l, Pr \\
\pi \cdot X \approx_\alpha{}^? t, Pr &\overset{[X/\pi^{-1}\cdot t]}{\Longrightarrow} Pr[X/\pi^{-1} \cdot t],\ \text{if } X \notin \mathtt{vars}(t) \\
t \approx_\alpha{}^? \pi \cdot X, Pr &\overset{[X/\pi^{-1}\cdot t]}{\Longrightarrow} Pr[X/\pi^{-1} \cdot t],\ \text{if } X \notin \mathtt{vars}(t)
\end{aligned}$$

Fig. 3. Simplification rules for unification problems

The rules from Fig. 3 induce a reduction relation on problems by "running the derivation rules in reverse", in no particular order. Write $Pr \Longrightarrow Pr'$ when $Pr'$ is obtained from $Pr$ by the application of a simplification rule from Fig. 3, and $\overset{*}{\Longrightarrow}$ for the transitive and reflexive closure of $\Longrightarrow$.

The nominal unification algorithm, called `unify` in the next sections, consists of applying the simplification rules to a problem $Pr = \langle \Delta, P \rangle$ until no more rules can be applied.

**Lemma 2.10 ([12])** *The relation $\Longrightarrow$ is strong normalizing and preserves solutions of problems, that is, $Pr \Longrightarrow Pr'$ implies $\mathcal{U}(Pr) = \mathcal{U}(Pr')$.*

We say that an equality problem $t \approx_\alpha^? u$ is *reduced* when one of the following holds:

(i) $t := a$ and $u := b$ are distinct atoms

(ii) Precisely one of $t$ and $u$ is a moderated variable and the other mentions that variable.

(iii) $t$ and $u$ have different term constructors at the root and neither is a variable.

(iv) $t$ and $u$ are applications with different term-formers.

We call a reduced equation, as above, *inconsistent*. Also, a freshness problem $a\#^? t$ is *reduced* if it is of the form $a\#^? X$ or $a\#^? a$. The former is called *consistent* and the latter *inconsistent*.

Normal forms are unique modulo renaming of variables, as in standard first-order unification. The normal form of a unification problem $Pr$ by $\Longrightarrow$ is defined as expected and denoted by $\langle Pr \rangle_{\mathtt{nf}}$. It consists of a set of equations and freshness constraints in reduced form. $Pr$ has a solution iff $\langle Pr \rangle_{\mathtt{nf}}$ contains only consistent reduced freshness constraints, i.e., freshness constraints of the form $a\#X$.

**Example 2.11** Consider the signature of lambda-calculus as in Example 2.2 and the problem below. We apply the simplification rules from Fig. 3 to get:

$$\begin{aligned}\{(\lambda\,[a]\,X)Z \approx_\alpha^? (\lambda\,[b]\,Y)b\} &\Longrightarrow \{\lambda\,[a]\,X \approx_\alpha^? \lambda\,[b]\,Y, Z \approx_\alpha^? b\}\\ &\overset{[Z/b]}{\Longrightarrow} \{\lambda\,[a]\,X \approx_\alpha^? \lambda\,[b]\,Y\}\\ &\Longrightarrow \{(b\ a)\cdot X \approx_\alpha^? Y, b\#^? X\}\\ &\overset{[Y/(b\ a)\cdot X]}{\Longrightarrow} \{b\#^? X\}\end{aligned}$$

Solution: $\langle b\#X, [Z/b, Y/(b\ a)\cdot X]\rangle$

### *2.4 Semantic Notions*

The semantics of solving equations is given by the nominal (universal) algebra. In [13], Gabbay gives an interpretation for nominal equational theories in nominal sets, constructs the *initial ground algebra* [3] $\mathbb{F}(T, \mathcal{D})$ for a nominal theory $T = (\Sigma, Ax)$ consisting of a signature $\Sigma$, a set of axioms $Ax$ of the form $\Delta \vdash s = t$ and a set of term-forms $\mathcal{D}$ disjoint from $\Sigma$, and proves a version of the HSP (Homomorphism, Subalgebra, Product) theorem for nominal algebras. Roughly speaking, the (HSP) theorem states that nominal equational varieties are closed under homomorphic images, subalgebras, products and atom abstraction. Basic definitions such as nominal sets, support, equivariance, and so on, are assumed and can be found in [17]. Below we give some semantic definitions and results taken from [15].

To start, derivations in a theory $T = (\Sigma, Ax)$ are defined by the rules in Figures 1 and 4. We say $\Pi$ is a *valid* derivation in $T$ when the following two conditions are satisfied:

[3] $\mathbb{F}(T, \mathcal{D})$ is called *free algebra* in [13]

- $\Pi$ mentions only terms in the signature $\Sigma$.
- $\Pi$ mentions only instances of $(ax_{\Delta' \vdash t=u})$ such that $(\Delta' \vdash t = u) \in Ax$.

$$\frac{}{\Delta \vdash t = t}\ (\text{refl}) \qquad \frac{\Delta \vdash t = u}{\Delta \vdash u = t}\ (\text{symm}) \qquad \frac{\Delta \vdash t = u \qquad \Delta \vdash u = v}{\Delta \vdash t = v}\ (\text{trans})$$

$$\frac{\Delta \vdash \pi(a)\#\pi \cdot X\sigma \text{ for every } a\#X \in \Delta'}{\Delta \vdash \pi \cdot t\sigma = \pi \cdot u\sigma}\ (ax_{\Delta' \vdash t=u})$$

$$\frac{\Delta \vdash t = u}{\Delta \vdash [a]t = [a]u}\ (\text{cong}[]) \qquad \frac{\Delta \vdash t = u}{\Delta \vdash f(\ldots, t, \ldots) = f(\ldots u \ldots)}\ (\text{cong}f)$$

$$\frac{\Delta, a\#X \vdash t = u\ (a \notin t, u)}{\Delta \vdash t = u}\ (\text{fr}) \qquad \frac{\Delta \vdash a\#t \qquad \Delta \vdash b\#t}{\Delta \vdash (a\ b) \cdot t = t}\ (\text{perm})$$

Fig. 4. Derivation rules for equality

To obtain the correctness of nominal equational logic, a relation between the syntax and semantics (as in Birkhoff's Theorem) will be established. For more details, we address the reader to [13], some extra concepts are included in the Appendix A.

For any nominal sets $\mathbb{X}$ and $\mathbb{Y}$ call a function $f \in \mathbb{X} \to \mathbb{Y}$ *equivariant* when $\pi \cdot f(x) = f(\pi \cdot x)$, for any $x \in \mathbb{X}$ and $\pi \in \mathbb{P}$.

**Definition 2.12** [$\Sigma$-algebra] A $\Sigma$-algebra $\mathcal{A}$ consists of:

- A domain nominal set $A = (A_S, \cdot)$, i.e., $A_S$ is a set equipped with a $\mathbb{P}$-group action $\cdot$ such that each $x \in A$ has finite support.
- An equivariant map $\texttt{atom} : \mathbb{A} \to A_S$ to interpret atoms; we write the interpretation $\texttt{atom}(a)$ as $a^{\mathcal{A}} \in A$.
- An equivariant map $\texttt{abs} : \mathbb{A} \times A_S \to A_S$ such that $a\#\texttt{abs}(a, x)$ always, to interpret abstraction.
- An equivariant map $f^{\mathcal{A}} : A_S^n \to A_S$ for each term-former $f : n \in \Sigma$ to interpret term-formers.

$\Sigma$-algebras are usually denoted by $\mathcal{A}, \mathcal{B}$.

As expected, a *valuation* $\varsigma$ in a $\Sigma$-algebra $\mathcal{A}$ maps unknowns $X$ to elements $\varsigma(X) \in A_S$. Below we define a equivariant function $[\![\cdot]\!]_\varsigma$ to interpret nominal terms w.r.t. a valuation $\varsigma$.

**Definition 2.13** Let $\mathcal{A}$ be a nominal algebra. Suppose that $t \in T(\Sigma, \mathbb{A}, \mathbb{X})$ and consider a valuation $\varsigma$ in $\mathcal{A}$. The interpretation $[\![t]\!]_\varsigma^{\mathcal{A}}$, or just $[\![t]\!]_\varsigma$ if $\mathcal{A}$ is understood, is defined inductively by:

$$[\![a]\!]_\varsigma = a^{\mathcal{A}} \quad [\![\pi \cdot X]\!]_\varsigma = \pi \cdot \varsigma(X) \quad [\![[a]\, t]\!]_\varsigma = \texttt{abs}(a, [\![t]\!]_\varsigma)$$
$$[\![f(t_1, \ldots, t_n)]\!]_\varsigma = f^{\mathcal{A}}([\![t_1]\!]_\varsigma, \ldots, [\![t_n]\!]_\varsigma)$$

We say a context interpretation $[\![\Delta]\!]_\varsigma^{\mathcal{A}}$ is *valid* when $a\#\varsigma(X)$ for each $a\#X \in \Delta$. In the same way, $[\![\Delta \vdash a\#t]\!]_\varsigma^{\mathcal{A}}$ is valid when $[\![\Delta]\!]_\varsigma^{\mathcal{A}}$ implies $a\#[\![t]\!]_\varsigma^{\mathcal{A}}$ and $[\![\Delta \vdash t = u]\!]_\varsigma^{\mathcal{A}}$

when $[\![\Delta]\!]^{\mathcal{A}}_{\varsigma}$ implies $[\![t]\!]^{\mathcal{A}}_{\varsigma} = [\![u]\!]^{\mathcal{A}}_{\varsigma}$.

A *model* of $T$ is a $\Sigma$-algebra $\mathcal{A}$ such that $[\![\Delta \vdash t = u]\!]_{\varsigma}$ is valid for every axiom $\Delta \vdash t = u$ in $Ax$ and every valuation $\varsigma$. If there is a derivation ending with $\Delta \vdash t = u$ that uses the axioms of $T$ and the derivation rules from Fig. 4 we write $\Delta \vdash_T t = u$.

To obtain models for a nominal equational theory $T$, the *initial ground algebra*, $\mathbb{F}(T, \mathcal{D})$ is built. As in [13], let $\mathcal{D}$ be a set of term-formers disjoint from $\Sigma$, they are called 'extra term-formers'. Then the set of *ground nominal terms* is generated by the grammar:

$$g ::= a \mid [a]\, g \mid f(g_1, \ldots, g_n) \mid d(a_1, \ldots, a_m)$$

Here $f : n$ ranges over elements of $\Sigma$ and $d : m$ ranges over elements of $\mathcal{D}$ (more details can be found in the Appendix).

Note that the initial ground (nominal) algebra is the analogous in the nominal setting to the initial first-order ground algebra $T(\Sigma)/E$, where $E$ is a set of equational axioms. The main difference here is that we add some new fresh term-formers from $\mathcal{D}$ to provide "enough ground terms" with non-empty support. A nominal algebra *variety* $\mathcal{V}$ for a signature $\Sigma$ is a collection of $\Sigma$-algebras closed under homomorphic images, subalgebras, countable products, and atom-abstractions. We say a collection $\mathcal{V}$ of $\Sigma$-algebras is *equational* when there is some theory $T = (\Sigma, Ax)$ such that $\mathcal{V}$ is the collection of all models of $T$.

**Theorem 2.14 (HSP Theorem, Theorem 9.3 in [13])** *A collection of $\Sigma$-algebras $\mathcal{V}$ is equational if, and only if, it is a variety.*

We also consider the term-algebra $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$, as in the first-order case, this term-algebra is *generic* (see [6]) for solving existentially closed equations.

**Corollary 2.15** *Let $\phi$ be the existentially closed equational judgment:*

$$\phi ::= \exists \overline{X}(\Delta \vdash s =_T t).$$

*Then $\Delta \vdash_{\mathbb{F}(T,\mathcal{D})} \phi$ if, and only if, $\Delta \vdash_{\mathcal{T}(\Sigma,\mathbb{A},\mathbb{X})} \phi$.*

**Example 2.16** Corollary 2.15 does not hold for disequations, to see this consider the theory $T = (\Sigma, Ax)$ where $\Sigma = \{f(\_)\}$ and $Ax = \{\vdash f(X) = a, \vdash [a]\, f(X) = a\}$. In $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$ one can derive $\vdash f(b) = a$ and $\vdash f(b) = b$:

$$\dfrac{\dfrac{}{\vdash id \cdot f(X)\sigma = a\sigma}\,(ax_{\Delta' \vdash f(X)=a}) \qquad \dfrac{}{\vdash (a\ b) \cdot f(X)\sigma' = (a\ b) \cdot a\sigma'}\,(ax_{\Delta' \vdash f(X)=a})}{\dfrac{\vdash f(b) = a \qquad \vdash f(b) = b}{\vdash a = b}\,(\text{trans})}$$

by taking $\sigma = \{X/b\}$ in the left branch and $\sigma' = \{X/a\}$ in the right branch. The dashed lines in the derivation above represent the result obtained after the application of substitutions $\sigma$ and $\sigma'$, and the swapping of names in the axiom $\vdash f(X) = a$. Therefore, every atom is in the same equivalence class modulo $T$.

Also, from the axioms of $T$ one can derive $\vdash f(t) = a$ and $\vdash [b]\, f(u) = a$ for any pair of terms $t$ and $u$. Hence it is possible to derive $\vdash t' = u'$ for any non-variable terms $t'$ and $u'$. The other equivalence classes are for variables $X, Y, Z \ldots$.

It follows that $\vdash_{\mathcal{T}(\Sigma,\mathbb{A},\mathbb{X})} \exists X.X \neq a$. However, $\mathbb{F}(T,\mathcal{D})$ (with $\mathcal{D}=\emptyset$) has only one equivalence class, i.e., the class of all ground terms and atoms, therefore, $\nvdash_{\mathbb{F}(T,\mathcal{D})} \exists X.X \neq a$ since every ground term and atom are in the same equivalence class modulo $T$.

## 3 Nominal Constraint Problems

In this section, we follow the approach proposed by Buntine and Bürckert [6] for solving a system of equations and disequations. Our approach, as in the first-order case, depends on the unification type of a (nominal) theory $T$. Fix the nominal algebra $\mathcal{T}(\Sigma,\mathbb{A},\mathbb{X})$ with the empty set of axioms, that is, terms are considered up to the built-in $\alpha$-equivalence. This theory is called CORE [15], and its deduction rules are given in Fig. 2. The results in this section can be extended to any theory $T$ provided unification is decidable and finitary for this theory; CORE has been chosen to make examples and proofs easier to follow.

**Definition 3.1** A *(nominal disunification) constraint problem* $\mathcal{P}$ is an ordered pair $\mathcal{P}=\langle E \mid\mid D\rangle$ where $E$ is a nonempty set of nominal equations-in-context $\Delta \vdash s \approx_\alpha t$ and $D$ is a (possible empty) set of nominal disequations-in-context $\nabla \vdash p \not\approx_\alpha q$, as follows:

$$E=\{\Delta_1 \vdash s_1 \approx_\alpha{}^? t_1, \cdots, \Delta_n \vdash s_n \approx_\alpha{}^? t_n\}$$
$$D=\{\nabla_1 \vdash p_1 \not\approx_\alpha{}^? q_1, \cdots, \nabla_m \vdash p_m \not\approx_\alpha{}^? q_m\}$$

The sets $\Delta_1,\ldots,\Delta_n,\nabla_1,\ldots,\nabla_m$ are consistent contexts. We call them the *initial freshness conditions* that are imposed on equations (disequations) in the problem $\mathcal{P}$.

In the case any of the $\Delta_i$ or $\nabla_j$ of a problem is empty we may write an equation(disequation)-in-context just as $s_i \approx_\alpha{}^? t_i$ ($p_j \not\approx_\alpha{}^? q_j$) instead of $\emptyset \vdash s \approx_\alpha{}^? t$. We also may consider the equations and disequations of the problems under the same context, $\Delta := \cup\Delta_i$ and $\nabla := \cup\nabla_j$.

**Remark 3.2** A constraint problem is equivalent to an existentially closed formula:

$$\mathcal{P} := \exists\overline{X}\left(\left(\bigwedge \Delta_i \vdash s_i \approx_\alpha t_i\right) \wedge \left(\bigwedge \nabla_j \vdash p_j \not\approx_\alpha q_j\right)\right).$$

We solve these formulas in the nominal term-algebra $\mathcal{T}(\Sigma,\mathbb{A},\mathbb{X})$ (see [13,15]), this is the logical task of finding witnesses/solutions for the variables in $\mathcal{P}$, that is, a pair $\langle\Gamma,\sigma\rangle$ where $\sigma$ is a substitution for the variables of the formula such that under some (possible empty) consistent context $\Gamma$ we have $\Gamma \vdash \mathcal{P}\sigma$.

To give some intuition on the construction of solution pairs, consider the constraint problem below:

$$\mathcal{P}=\left\langle (b\ a)\cdot X \approx_\alpha{}^? Y \mid\mid [a]\,X \not\approx_\alpha{}^? [b]\,Y \right\rangle \tag{1}$$

The intended effect of a solution $\langle\Gamma,\sigma\rangle$ of $\mathcal{P}$ is that it needs to solve the equation $(b\ a)\cdot X \approx_\alpha{}^? Y$ and the disequation $[a]\,X \not\approx_\alpha{}^? [b]\,Y$ where solving this disequation

means $\Gamma \not\vdash [a]\,X\sigma \approx_\alpha [b]\,Y\sigma$, i.e., $\langle \Gamma, \sigma\rangle$ is not a solution of the equation $[a]\,X \approx_\alpha{}^? [b]\,Y$ which will be called the *associated equation* to the problem $[a]\,X \not\approx_\alpha{}^? [b]\,Y$. Notice that,

$$\langle \Gamma, \sigma\rangle = \langle \emptyset, [Y/(b\ a)\cdot X]\rangle \tag{2}$$

solves the constraint problem $\mathcal{P}$ above. The main goal of this section is show how to construct these solutions.

In general, instantiation plays an important role in unification theory. It is by instances of more general unifiers (instantiation closure) that one produces a finite representation of all other solutions of a unification problem. Therefore, it is helpful to have the property of instantiation closure to solutions of constraints problems as well. Unfortunately, this is not the case since we are solving constraints in the nominal term-algebra $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$. For an example, let $\mathcal{Q}$ be the constraint problem:

$$\mathcal{Q} = \left\langle X \approx_\alpha{}^? Y \;||\; X \not\approx_\alpha{}^? a \right\rangle$$

The pair $\langle \Gamma, \sigma\rangle = \langle \emptyset, [X \mapsto (a\ b)\cdot Z, Y \mapsto (a\ b)\cdot Z]\rangle$ solves $\mathcal{Q}$. However, if we instantiate this solution with $\delta = [Z \mapsto b]$ the instance $\langle \emptyset, [X/a, Y/a]\rangle$ is not a solution of $\mathcal{Q}$.

**Example 3.3** Let $\langle \Gamma, \sigma\rangle = \langle \emptyset, [Y/(b\ a)\cdot X]\rangle$, as in (2). Consider the pair $\langle \Gamma', \sigma\rangle = \langle b\#X, [Y/(b\ a)\cdot X]\rangle$. Notice that $\langle \Gamma, \sigma\rangle \leq \langle \Gamma', \sigma\rangle$, therefore $\langle \Gamma', \sigma\rangle$ solves $(b\ a)\cdot X \approx_\alpha{}^? Y$. In addition, $\langle \Gamma', \sigma\rangle$ is a solution of the equation $[a]\,X \approx_\alpha{}^? [b]\,Y$ associated to $[a]\,X \not\approx_\alpha{}^? [b]\,Y$. It can not solve (1) since it solves the equations and the associated equation $[a]\,X \approx_\alpha{}^? [b]\,Y$.

The reader may wonder if such an anomaly is caused by the context assumptions added on the initial problem or, reasonable enough, even the $\alpha$-equivalence embedded in the theory of nominal terms. Certainly, context assumptions seem to cause some difficulties. Firstly, because that the notion of instantiation may introduce new freshness constraints, as in Example 3.3. Secondly, freshness conditions on the equational part of a constraint problem can interact with solutions and, as showed in the example below, even change the solvability of a problem.

**Example 3.4** Consider the following modification of the original problem (1):

$$\mathcal{P}' = \left\langle b\#X \vdash (b\ a)\cdot X \approx_\alpha{}^? Y \;||\; [a]\,X \not\approx_\alpha{}^? [b]\,Y \right\rangle$$

Notice that $\mathcal{P}'$ does not have a solution: every time we solve $b\#X \vdash (b\ a)\cdot X \approx_\alpha{}^? Y$ we always solve the equation $[a]\,X \approx_\alpha{}^? [b]\,Y$ associated to $[a]\,X \not\approx_\alpha{}^? [b]\,Y$.

We will work on this type of issues in the remaining of this paper. First, we define precisely what we mean by a solution of nominal constraint problems. Keep in mind that our goal is the development of a nominal generalization for the already established notion of instantiation of solutions (Definition 2.8), but this needs to be done in such a way that instantiation closure still holds.

### 3.1 Generalized instantiation

In this subsection, some notions initially established in [6] will be extended into the nominal framework. The main difference is the lifting of the notion of *substitution with exceptions* to pairs of the form $\langle \Gamma, \sigma \rangle$ consisting of a consistent freshness context and a substitution, in addition of course, to the fact that $\alpha$-equality is axiomatized in nominal terms which adds some complexity when compared to syntactic equality. Besides, we have adapted the Consistency Test Algorithm (Algorithm 1) to deal with pairs with exceptions.

**Definition 3.5** A *pair with exceptions*, denoted as $\langle \Gamma, \sigma \rangle - \Psi$, consists of a pair $\langle \Gamma, \sigma \rangle$ and an indexed family of the form $\Psi = \{\langle \nabla_l, \psi_l \rangle \mid l \in I\}$.

*Pairs with exceptions* will be used as a representation of solutions of a constraint problem that has restrictions on how they can be instantiated. For instance, in the problem $\mathcal{Q}$ above, solutions of the equation $X \approx_\alpha Y$ can be instantiated in any way *except* for the instances where $X$ is mapped to $a$.

**Definition 3.6** [Pair instances]

- A pair $\langle \Gamma, \sigma \rangle$ is said to be an *instance of a family* $\Psi = \{\langle \nabla_l, \psi_l \rangle \mid l \in I\}$, denoted by $\Psi \leq \langle \Gamma, \sigma \rangle$, if and only if each instance of $\langle \Gamma, \sigma \rangle$ is an instance of some $\langle \nabla_l, \psi_l \rangle$ in $\Psi$.
- A pair $\langle \Delta, \lambda \rangle$ is an *instance of a pair with exceptions* $\langle \Gamma, \sigma \rangle - \Psi$, written $\langle \Gamma, \sigma \rangle - \Psi \leq \langle \Delta, \lambda \rangle$, if and only if $\langle \Delta, \lambda \rangle$ is an instance of $\langle \Gamma, \sigma \rangle$ but not of $\Psi$.

**Definition 3.7** A pair with exceptions $\langle \Gamma, \sigma \rangle - \Psi$ is consistent if and only if it has at least one instance.

For example, the pair with exceptions $\langle b\#X, [Y/(b\ a)\cdot X] \rangle - \{(b\#X, [Y/(b\ a)\cdot X])\}$ from Example 3.4 is inconsistent. The following lemma is a useful characterization of consistency for pair with exceptions.

**Lemma 3.8 (Inconsistency Lemma)** *A pair with exceptions $\langle \Gamma, \sigma \rangle - \Psi$ is inconsistent if and only if $\langle \Gamma, \sigma \rangle$ is an instance of $\Psi$.*

**Proof.** ($\Rightarrow$) If $\langle \Gamma, \sigma \rangle$ is an instance of $\Psi$ then all instances $\langle \Delta, \gamma \rangle \leq \langle \Gamma, \sigma \rangle$ is an instance of some $\langle \nabla_i, \psi_i \rangle$ in $\Psi$, so by Definition 3.6 $\langle \Gamma, \sigma \rangle - \Psi$ has no instances hence it is inconsistent.

($\Leftarrow$) Conversely, suppose $\langle \Gamma, \sigma \rangle - \Psi$ is consistent and $\langle \Gamma, \sigma \rangle$ is an instance of $\Psi$. Then there exists an instance $\langle \Delta, \lambda \rangle$ of $\langle \Gamma, \sigma \rangle - \Psi$. Hence $\langle \Gamma, \sigma \rangle \leq \langle \Delta, \lambda \rangle$. Since $\langle \Gamma, \sigma \rangle$ is an instance of $\Psi$ we have $\langle \nabla_i, \psi_i \rangle \leq \langle \Gamma, \sigma \rangle$, by transitivity

$$\langle \nabla_i, \psi_i \rangle \leq \langle \Gamma, \sigma \rangle \leq \langle \Delta, \lambda \rangle$$

a contradiction with Definition 3.6. □

Recalling Definition 2.13, we say that a pair with exceptions $\langle \Gamma, \sigma \rangle - \psi$ is inconsistent on a $\Sigma$-algebra $\mathcal{A}$ iff $\mathit{instances}(\langle \Gamma, \sigma \rangle - \Psi) = \emptyset$ in $\mathcal{A}$, where $\mathit{instances}(\langle \nabla, \rho \rangle) = \{\langle \nabla, \rho \rangle \varsigma \mid \text{ for all valuation } \varsigma\}$.

**Corollary 3.9** *If $\langle \Gamma, \sigma\rangle - \psi$ is inconsistent on $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$ then it is inconsistent on the ground algebra $\mathbb{F}(CORE, \mathcal{D})$.*

**Proof.** If $\langle \Gamma, \sigma\rangle - \psi$ is inconsistent on $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$ then each instance $\langle \Gamma', \sigma'\rangle$ of $\langle \Gamma, \sigma\rangle$ is in turn an instance of some $\langle \Delta_l, \psi_l\rangle \in \Psi$, i.e., in terms of Definition 2.8, there exists $\delta$ such that

$$\text{for all } X \in \mathbb{X}, \qquad \Gamma' \vdash_{\mathcal{T}(\Sigma,\mathbb{A},\mathbb{X})} X\sigma' = X\psi_l\delta \qquad \text{and} \qquad \Gamma' \vdash \Delta_l\delta.$$

The result follows from Corollary 2.15. Notice that the converse is not true in general. For instance, consider the theory $T$ as in Example 2.16, all pair with exceptions are inconsistent on $\mathbb{F}(T, \mathcal{D})$ since it has only one equivalence class but this not happens in $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$. □

**Corollary 3.10** *Let $\langle \Gamma, \sigma\rangle - \Psi$ be a pair with exceptions. If there is some $\langle \nabla_l, \psi_l\rangle \in \Psi$ such that there exists a substitution $\delta$ satisfying*

$$\Gamma \vdash X\sigma \approx_\alpha X\psi_l\delta, \text{ for all } X \in \mathtt{vars}(\mathcal{P}).$$

*Then $\langle \Gamma, \sigma\rangle - \Psi$ is inconsistent if and only if $\Gamma \supseteq \langle \nabla_l\delta\rangle_{\mathtt{nf}}$.*

**Proof.** Consider the pair with exceptions $\langle \Gamma, \sigma\rangle - \Psi$ as above. From assumption, $\sigma$ is an instance of $\psi_l$ over the context $\Gamma$. By the inconsistency lemma this pair with exceptions is inconsistent iff $\langle \Gamma, \sigma\rangle$ is an instance of $\Psi$. The result follows from the fact that $\Gamma \vdash \nabla_l\delta$ iff $\Gamma \supseteq \langle \nabla_l\delta\rangle_{\mathtt{nf}}$. □

The above corollary enables us to algorithmically test if some pair with exceptions is consistent provided that we have already solved the matching-in-context problem $(\Gamma \vdash X\sigma) \approx_? (\Gamma \vdash X\psi)$ (for all variables $X$ appearing in the constraint problem) where $X\sigma$ is the pattern (see [12, Definition 45]). That is, for each $\psi_l$ we solve the unification problem

$$\Gamma \vdash X_1\sigma \approx_\alpha{}^? X_1\psi_l, \cdots, X_n\sigma \approx_\alpha{}^? X_n\psi_l$$

without instantiating variables of $X_i\sigma$, for all $1 \leq i \leq n$. The solution of this matching problem (if it exists) will be denoted by $\delta$. In [7], the authors give an efficient implementation for the matching problem.

## 4 Solving nominal constraints

Finally, we give the formal definition of a solution of a nominal constraint problem and also construct a finite representation for the solution set.

**Definition 4.1** Let $\mathcal{P} = \langle \Delta \vdash s_1 \approx_\alpha{}^? t_1, \cdots, s_n \approx_\alpha{}^? t_n \,||\, \nabla \vdash p_1 \not\approx_\alpha{}^? q_1, \cdots, p_m \not\approx_\alpha{}^? q_m\rangle$ be a nominal disunification constraint problem. A solution of $\mathcal{P}$ is a pair $\langle \Gamma, \sigma\rangle$ of a consistent context $\Gamma$ and a substitution $\sigma$ satisfying the following conditions:

(i) $\langle \Gamma, \sigma\rangle$ is a solution of the equational part $E$ of $\mathcal{P}$.

**Algorithm 1** *Consistency Test*

**input:** $\langle \Gamma, \sigma \rangle - \psi$ *a finite pair with exceptions.*
**output:** `true` *if the input is consistent* `false` *otherwise.*
**for each** $\langle \nabla_l, \psi_l \rangle \in \Psi$ **do**
  **if** $matching(\Gamma, X_1\sigma \approx_? X_1\psi_l, \cdots, X_n\sigma \approx_? X_n\psi_l) = \delta$ **then**
    **if** $\Gamma \supseteq \langle \nabla_l \delta \rangle_{\mathtt{nf}}$ **then**
      **return** `false` *and* **stop**
    **end**
  **end**
**end**
**return** `true`

(ii) $\langle \Gamma, \sigma \rangle$ satisfies the disequations in the disequational part $D$ of $\mathcal{P}$, that is:
  (a) $\Gamma \nvdash \nabla\sigma$, or
  (b) $\Gamma \nvdash p\sigma \approx_\alpha q\sigma$, for all $p \not\approx_\alpha{}^? q$ in $D$.

**Algorithm 2** *Construction of a complete representation of solutions of constraint problems*

**input:** *A disunification problem* $\mathcal{P} = \langle E \mid\mid D \rangle$.
**output:** *A finite set* $S$ *of pairs with exceptions (possibly empty).*
**let** $\langle \Gamma, \sigma \rangle := \mathtt{unify}(E)$
**let** $\Psi := \bigcup_{p_i \not\approx_\alpha{}^? q_i \in D} \{\langle \nabla_i, \psi_i \rangle = \mathtt{unify}(\nabla_i, p_i \approx_\alpha{}^? q_i)\}$
**if** $\mathtt{consistent}(\langle \Gamma, \sigma \rangle - \Psi)$ **then**
  **return** $\langle \Gamma, \sigma \rangle - \Psi$
**else**
  **return** $\emptyset$
**end**

**Definition 4.2** We call a set $S$ of pairs with exceptions a *complete representation* of the solutions of the constraint problem $\mathcal{P}$ iff $S$ satisfies the following conditions:

(i) If $\langle \Gamma, \sigma \rangle - \Psi \leq (\Delta, \lambda)$ for some $\langle \Gamma, \sigma \rangle - \Psi$ in $S$ then $\langle \Delta, \lambda \rangle$ solves $\mathcal{P}$.
(ii) If $\langle \Delta, \lambda \rangle$ solves $\mathcal{P}$ then it is an instance of some $\langle \Gamma, \sigma \rangle - \Psi$ in $S$.
(iii) $\langle \Gamma, \sigma \rangle - \Psi$ is consistent for all $\langle \Gamma, \sigma \rangle - \Psi \in S$.

Similar to nominal unification problems, we are interested in generating a complete finite representation for the set of solutions to a constraint problem $\mathcal{P}$. We use Algorithm 2 to compute such a representation in the form of a pair with exceptions $\langle \Gamma, \sigma \rangle - \Psi$ where $\langle \Gamma, \sigma \rangle$ is a solution for the equations in $\mathcal{P}$ and the family $\Psi = \{\langle \Delta_l, \psi_l \rangle\}$ is formed by taking each pair $\langle \Delta_l, \psi_l \rangle$ as the solution of the associated equations $\Delta \vdash p_l \approx_\alpha{}^? q_l$, $1 \leq l \leq m$. Termination of Algorithm 2 follows from the termination of `unify`, and correctness (soundness and completeness) follows from the Representation Theorem below.

**Theorem 4.3 (Representation Theorem)** *Let*

$$\mathcal{P} = \left\langle \Delta \vdash s_1 \approx_\alpha^? t_1, \cdots, s_n \approx_\alpha^? t_n \;||\; \nabla \vdash p_1 \not\approx_\alpha^? q_1, \cdots, p_m \not\approx_\alpha^? q_m \right\rangle$$

*be a nominal constraint problem. Define the family*

$$\Psi := \bigcup_{p \not\approx_\alpha^? q \in D} \mathcal{U}\left(\nabla, p \approx_\alpha^? q\right).$$

*Then the set* $S = \{\langle \Gamma, \sigma\rangle - \Psi \mid \langle \Gamma, \sigma\rangle \in \mathcal{U}(E)$ *and* $\Psi \not\leq \langle \Gamma, \sigma\rangle\}$ *is a complete representation of solutions for the constraint problem* $\mathcal{P}$.

**Proof.**

(i) Take $\langle \Lambda, \lambda\rangle$ an instance of some $\langle \Gamma, \sigma\rangle - \Psi$ in $S$. Then $\langle \Gamma, \sigma\rangle \leq \langle \Lambda, \lambda\rangle$ and it is not an instance of $\Psi$. Since unification problems are closed by instantiation it follows that $\langle \Lambda, \lambda\rangle$ solves the equational part of $\mathcal{P}$. It remains to show that $\langle \Lambda, \lambda\rangle$ solves the disequational part of $\mathcal{P}$. Suppose by contradiction that $\langle \Lambda, \lambda\rangle$ satisfies $\nabla \vdash p_l \approx_\alpha q_l$ for some $\nabla \vdash p_l \not\approx_\alpha q_l$ in $D$. Therefore, $\langle \Lambda, \lambda\rangle$ is an instance of $\langle \nabla_l, \psi_l\rangle$ (a solution of the associated unification problem $\nabla \vdash p_l \approx_\alpha^? q_l$ in $D$) and every instance of $\langle \Lambda, \lambda\rangle$ is an instance of $\langle \nabla_l, \psi_l\rangle$ then $\Psi \leq \langle \Lambda, \lambda\rangle$, a contradiction.

(ii) Suppose $\langle \Lambda, \lambda\rangle$ solves $\mathcal{P}$. Then, $\langle \Lambda, \lambda\rangle$ solves the equational (disequational) part of $\mathcal{P}$. Consider $\langle \Gamma, \sigma\rangle \in \mathcal{U}(E)$ a solution of $E$, then we conclude that $\langle \Gamma, \sigma\rangle \leq \langle \Lambda, \lambda\rangle$. In addition, $\langle \Lambda, \lambda\rangle$ solves the disequational part of $\mathcal{P}$ as well, that is;

$$\Lambda \nvdash \nabla\lambda \qquad \text{or} \qquad \Lambda \nvdash p\lambda \approx_\alpha q\lambda, \quad \text{for all } p \not\approx_\alpha^? q \in D \tag{3}$$

Assume $\langle \Lambda, \lambda\rangle$ is an instance of $\Psi$. Then all instances of $\langle \Lambda, \lambda\rangle$ is an instance of some $\langle \nabla_l, \psi_l\rangle$ in $\Psi$. Hence, there is some $\langle \nabla_l, \psi_l\rangle$ in $\Psi$ such that

$$\langle \nabla_l, \psi_l\rangle \leq \langle \Lambda, \lambda\rangle.$$

A contradiction with (3). Therefore, $\langle \Lambda, \lambda\rangle$ can not be an instance of $\Psi$ and, we conclude that $\langle \Lambda, \lambda\rangle$ is an instance of $\langle \Gamma, \sigma\rangle - \Psi$, as required.

□

**Remark 4.4**

(i) Note that any ground instance of a pair with exception representing a solution of a constraint problem $\mathcal{P}$ is also a solution of $\mathcal{P}$. We can restrict solutions to ground instances, but this does not mean that if a problem is solvable in the term-algebra $\mathcal{T}(\Sigma, \mathbb{A}, \mathbb{X})$ it is also solvable in the ground algebra $\mathbb{F}(CORE, \mathcal{D})$, as discussed earlier in Example 2.16.

(ii) If one wants to solve a disunification problem in the initial ground algebra, by Lemma 3.8, one needs to test if all ground instances of the solutions to

the equational part $E$ are an instance of the exceptions $\psi$. For some nominal theories this is not an easy task.

(iii) We have a restricted instantiation closure, it is not transitive. In fact, $\langle \emptyset, [X/Z] \rangle$ is an instance of the pair with exception $\langle \emptyset, [X/Y] \rangle - \langle \emptyset, [X/a] \rangle$. Note that $\langle \emptyset, [X/Z] \rangle \leq \langle \emptyset, [X/a] \rangle$ but the latter is not an instance of $\langle \emptyset, [X/Y] \rangle - \langle \emptyset, [X/a] \rangle$.

**Example 4.5** Consider the constraint problem $\mathcal{P}$ below:

$$\mathcal{P} = \left\langle \lambda\,[a]\,X \approx_\alpha{}^? \lambda\,[b]\,Y \;||\; X \not\approx_\alpha{}^? Y, X \not\approx_\alpha{}^? a \right\rangle.$$

First apply `unif` to the equational part of the problem obtaining as result:

$$\langle \Gamma, \sigma \rangle = \langle b\#X, [Y/(b\ a) \cdot X] \rangle \tag{4}$$

Then solve the associated equations of the disequational part to combine them as a family of pairs with exception:

$$\Psi = \{\langle \emptyset, [X/Y] \rangle, \langle \emptyset, [X/a] \rangle\} \tag{5}$$

Finally form the pair with exception $\langle \Gamma, \sigma \rangle - \Psi$ by the combination of (4) and (5). We can check consistency of $\langle \Gamma, \sigma \rangle - \Psi$ using Algorithm (1).

## 5 Related Work

Disunification problems have been studied extensively in the first-order framework [9,10,6,11,5,18] and also in the higher-order one [16].

Buntine and Bürckert [6] solve systems of equations and disequations in equational theories with a finitary unification type; they investigate $E$-disunification problems with two main applications in mind: the first application is to give a generalization for logic programming to include negation clauses in such a way that solution to queries can be expressed as substitutions other than the limited form of negation, called negation as failure.

The second applications is related with the use of $E$-disunification as a mechanism to drastically reduce the solution space of the unification algorithm for some equational theories. For instance, they showed that associative-commutative unification problems (a.k.a. $AC$-unification problems) are in fact a kind of so called $AC1$-disunification problems (associative-commutative functions with a unity 1) that have a solution space considerably smaller than the solution space of standard $AC$-problems. Differently, Comon and Lescanne [10,9] consider more general problems, called *equational problems*, which include universally and existentially quantified variables in the algebra of rational trees or in the quotient term-algebra $T(F, X)$ by a congruence $=_E$. They propose a set of transformation rules on equational problems of the form $\exists \overline{w} \forall \overline{y} : P_1 \wedge \ldots \wedge P_n$, where $P_i$, for $i = 1..n$, is a called a system, that is, an equation of the form $s = t$ or $\top$, or a disequation $s \neq t$ or $\bot$, or a disjunction $P_{i1} \vee \ldots \vee P_{in_i}$ of systems. Their strategy consists of applying transformation rules

to the equational problem until a kind of *solved form* is reached. These problems have applications in *sufficient completeness* for algebraic specifications defined by sets of rewriting rules.

In [11], Fernández shows that $E$-disunification is semi-decidable when the theory $E$ is presented by a ground convergent rewrite system, and gives a sound and complete $E$-disunification procedure based on narrowing. Baader and Schulz [5] show that solvability of disunification problems in the free algebra of the combined theory $E_1 \cup \ldots \cup E_n$ is decidable if solvability of disunification problems with linear constant restrictions in the free algebras of the theories $E_i (1 \leq i \leq n)$ is decidable. Lugiez [16] introduces higher-order disunification problems and gives some decidable cases for which equational problems can be extended to higher-order systems.

## 6 Conclusions and Future Work

In this work, we have provided a method to deal with nominal equations constrained by equality constraints in the form of nominal disequations. The approach adapts Buntine and Bürckert's first-order method to solve disequations taking into account the particularities of nominal syntax and semantics. To the best of our knowledge, this is the first work that deals with disequations in the nominal setting. The main result, Theorem 4.3, establishes the soundness and completeness of the proposed approach.

As future work, we intend to investigate more specific applications of nominal constraint problems; inspired from Buntine and Bürkert's work we could seek some direct extensions to nominal logic programming with negated equations and apply our results to more general unification theories (for instance, $AC$ and $AC1$-nominal unification problems). Also, the more general approach to disunification followed by Comon and Lescane [10] using quantified variables will be investigated.

## References

[1] Ayala-Rincón, M., W. de Carvalho Segundo, M. Fernández and D. Nantes-Sobrinho, *On solving nominal fixpoint equations*, in: *Proc. 11th International Symposium on Frontiers of Combining Systems, FroCoS*, Lecture Notes in Computer Science **10483** (2017), pp. 209–226.
URL https://doi.org/10.1007/978-3-319-66167-4_12

[2] Ayala-Rincón, M., W. de Carvalho Segundo, M. Fernández and D. Nantes-Sobrinho, *Nominal c-unification*, in: *27th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2017, Revised Selected Papers*, Lecture Notes in Computer Science **10855** (2018), pp. 235–251.
URL https://doi.org/10.1007/978-3-319-94460-9_14

[3] Ayala-Rincón, M., M. Fernández and D. Nantes-Sobrinho, *Nominal narrowing*, in: D. Kesner and B. Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*, LIPIcs **52** (2016), pp. 11:1–11:17.
URL https://doi.org/10.4230/LIPIcs.FSCD.2016.11

[4] Ayala-Rincón, M., M. Fernández and D. Nantes-Sobrinho, *Fixed-point constraints for nominal equational unification*, in: H. Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, LIPIcs **108** (2018), pp. 7:1–7:16.
URL https://doi.org/10.4230/LIPIcs.FSCD.2018.7

[5] Baader, F. and K. U. Schulz, *Combination techniques and decision problems for disunification*, Theor. Comput. Sci. **142** (1995), pp. 229–255.
URL https://doi.org/10.1016/0304-3975(94)00277-0

[6] Buntine, W. L. and H.-J. Bürckert, *On solving equations and disequations*, J. ACM **41** (1994), pp. 591–629.
URL http://doi.acm.org/10.1145/179812.179813

[7] Calvès, C. and M. Fernández, *Matching and alpha-equivalence check for nominal terms*, Journal of Computer and System Sciences **76** (2010), pp. 283 – 301.
URL https://doi.org/10.1016/j.jcss.2009.10.003

[8] Cheney, J., *Equivariant unification*, Journal of Automated Reasoning **45** (2010), pp. 267–300.
URL https://doi.org/10.1007/s10817-009-9164-3

[9] Comon, H., *Disunification: a Survey*, in: J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, MIT Press, 1991 pp. 322–359.

[10] Comon, H. and P. Lescanne, *Equational problems and disunification*, Journal of Symbolic Computation **7** (1989), pp. 371 – 425, unification: Part 1.
URL http://www.sciencedirect.com/science/article/pii/S0747717189800173

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

[12] Fernández, M. and M. J. Gabbay, *Nominal rewriting*, Information and Computation **205** (2007), pp. 917 – 965.
URL https://doi.org/10.1016/j.ic.2006.12.002

[13] Gabbay, M. J., *Nominal Algebra and the HSP Theorem*, Journal of Logic and Computation **19** (2008), pp. 341–367.
URL https://doi.org/10.1093/logcom/exn055

[14] Gabbay, M. J. and A. Mathijssen, *The lambda-calculus is nominal algebraic*, in: C. Benzmüller, C. Brown, J. Siekmann and R. Statman, editors, *Reasoning in simple type theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, Studies in Logic and the Foundations of Mathematics **17**, College Publications, 2008 pp. 271–302.
URL http://www.gabbay.org.uk/papers/lamcna.pdf

[15] Gabbay, M. J. and A. Mathijssen, *Nominal (Universal) algebra: equational logic with names and binding* **19** (2009), pp. 1455–1508.
URL https://doi.org/10.1093/logcom/exp033

[16] Lugiez, D., *Higher order disunification: Some decidable cases*, in: *First International Conference on Constraints in Computational Logics, CCL*, Lecture Notes in Computer Science **845** (1994), pp. 121–135.
URL https://doi.org/10.1007/BFb0016848

[17] Pitts, A. M., "Nominal Sets: Names and Symmetry in Computer Science," Cambridge University Press, New York, NY, USA, 2013.

[18] Ravishankar, V., K. A. Gero and P. Narendran, *Asymmetric unification and disunification*, CoRR **abs/1706.05066** (2017).
URL http://arxiv.org/abs/1706.05066

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

# A More on Nominal Semantics

As in [13], let $\mathcal{D}$ be a set of term-formers disjoint from $\Sigma$, they are called 'extra term-formers'. Then the set of *ground nominal terms* is generated by the grammar:

$$g ::= a \mid [a]\, g \mid f(g_1, \ldots, g_n) \mid d(a_1, \ldots, a_m)$$

Here $f : n$ range over elements of $\Sigma$ and $d : m$ ranges over elements of $\mathcal{D}$ (More details can be found in the Appendix).

**Definition A.1** If $g \in \mathbb{F}(\Sigma, \mathcal{D})$ write $[g]_T$ for the set of $g' \in \mathbb{F}(\Sigma, \mathcal{D})$ such that there exists a derivation $\vdash_T g = g'$. Write $\mathbb{F}(T, \mathcal{D})$ for the nominal set such that:

- $\mathbb{F}(T,\mathcal{D})_S = \{[g]_T \mid g \in \mathbb{F}(\Sigma,\mathcal{D})\}$ is the underlying set.
- $\pi \cdot [g]_T = [\pi \cdot g]_T$.

**Definition A.2** Let $T = (\Sigma, Ax)$ and suppose $\mathcal{D}$ is a set of term-formers disjoint from $\Sigma$. The *initial ground algebra* of $T$ over $\mathcal{D}$, denoted by $\mathbb{F}(T,\mathcal{D})$, is the $\Sigma$-algebra with:

- Underlying nominal set $\mathbb{F}(T,\mathcal{D})_S$, as defined in Definition A.1.
- Interpretation of atoms defined as $a^{\mathcal{A}} = [a]_T$
- Interpretation of abstraction $\mathtt{abs}(a,x) = [[a]\, g]_T$ for some $g \in x$.
- $f^{\mathcal{A}}(x_1,\ldots,x_n) = [f(g_1,\ldots,g_n)]_T$ for some $g_1 \in x_1,\ldots,g_n \in x_n$, for each term-former $f : n \in \Sigma$.

For $\Sigma$-algebras $\mathcal{A}$ and $\mathcal{B}$, a $\Sigma$-algebra homomorphism from $\mathcal{A}$ to $\mathcal{B}$ is an equivariant function $A_S \to B_S$ that preserves the interpretation for atoms, abstraction, and function symbols. If $\phi : \mathcal{A} \to \mathcal{B}$ is a surjective homomorphism onto $\mathcal{B}$ then we say $\mathcal{B}$ is a homomorphic image of $\mathcal{A}$.

**Theorem A.3** *For any nominal equational theory $T$, $\mathbb{F}(T,\mathcal{D})$ is a $\Sigma$-algebra and $\mathbb{F}(T,\mathcal{D})$ is a model of $T$.*

Theorems A.4 and A.5 are technical and used in the proof of the HSP theorem. The first states that every element of a variety $\mathcal{V}$ is a homomorphic image of some sufficiently large initial ground algebra. The second states that given a collection $\mathcal{V}$ of $\Sigma$-algebras, the theory $T$ generated by $\mathcal{V}$ induces an injective $\Sigma$-algebra homomorphism $\theta : \mathbb{F}(T,\mathcal{D}) \to \Pi_{A_i \in \mathcal{V}} \mathcal{A}_i$, into the product algebra $\Pi_{A_i \in \mathcal{V}} \mathcal{A}_i$.

**Theorem A.4** *Fix a signature $\Sigma$, a nominal theory $T$, and $\mathcal{V}$, the variety of $T$. If $\mathcal{A} \in \mathcal{V}$ then the exists some (sufficiently large) set of term-formers $\mathcal{D}$ such that there exists a $\Sigma$-algebra homomorphism $\theta$ from $\mathbb{F}(T,\mathcal{D})$ to $\mathcal{A}$ such that $\theta$ is surjective.*

**Theorem A.5** *Suppose $\mathcal{V}$ is a collection of $\Sigma$-algebras. Let $T$ be the theory generated by $\mathcal{V}$. Suppose $\mathcal{D}$ is any set of term-formers, so $(\Sigma \cap \mathcal{D}) = \emptyset$. Then there exists some indexing set $I$ and algebras $\mathcal{A}_i \in \mathcal{V}$ for $i \in I$ such that there exists a $\Sigma$-algebra homomorphism $\theta$ from $\mathbb{F}(T,\mathcal{D})$ to $\Pi_{i \in I} \mathcal{A}_i$ such that $\theta$ is injective.*