Nominal Disunification via Fixed-Point Constraints (Work in Progress)

Leonardo M. Batista[2,*], Maribel Fernández[1], Daniele Nantes-Sobrinho[2], and Deivid Vale[3]

[1] King's College London, London, UK
maribel.fernandez@kcl.ac.uk
[2] Universidade de Brasília, Brasília, Brazil
L.M.Batista@mat.unb.br, dnantes@unb.br
[3] Radboud University Nijmegen, Nijmegen, The Netherlands
deividvale@cs.ru.nl

Abstract

This is a work in progress on solving equations and disequations between nominal terms, i.e, we are interested in the *nominal disunification problem*. In the standard nominal syntax, α-equality ($s \approx_\alpha t$) between two nominal terms s and t is defined using a freshness predicate ($a\#t$) meaning "atom a is fresh for t". Recently, an alternative syntax was proposed using fixed-point constraints instead of freshness constraints. Using fixed-point constraints, nominal commutative ($\mathtt{C}$) unification is finitary whilst it is not finitary if freshness constraints are used to represent solutions. With the (future) goal of investigating nominal disunification problems modulo equational theories, whose solvability via freshness constraints may be problematic, we exploit this fixed-point approach to solve nominal disunification problems: we provide an algorithm to compute finite and complete sets of solutions for nominal disunification problems consisting of equations, disequations and fixed-point constraints. This is a first step towards solving nominal $\mathtt{C}$-disunification problems.

1 Introduction

This paper is about solving equations ($s \approx_\alpha^? t$) and disequations ($s \not\approx_\alpha^? t$) between nominal terms, that is, it concerns the *nominal disunification problem* [3], which has the form $\langle s_i \approx_\alpha^? t_i \ (1 \leq i \leq n), u_j \not\approx_\alpha^? v_j \ (1 \leq j \leq m)\rangle$.

Nominal techniques are useful for the treatment of languages involving binders [6]. In this approach, bindings are implemented through the abstraction of atoms, and atom permutations are used to implement renamings. For this, freshness constraints (which have the form $a\#t$) and α-equivalence constraints (which have the form $s \approx_\alpha t$) are considered. Intuitively, $a\#t$ means that the atom a cannot occur free in the term t. This concept was formalised in [7] using the quantifier *new* (И) which, in nominal logic, quantifies over new names. Such formalisation is expressed by the following sentence: $a\#x \Leftrightarrow (\text{И}a')(a\ a') \cdot x = x$, that is, a is fresh in x if, and only if, for any new atom a', the permutation $(a\ a')$ fixes x. For example, consider the formula $\phi = \forall[a]P$. In this case, a is an abstracted name, therefore $a\#\phi$, which is equivalent to saying that the renaming of a by a new name a' still preserves ϕ, that is, $(\text{И}a')(a\ a') \cdot \phi \approx_\alpha \phi$.

This observation lead to a new axiomatisation of α-equivalence of nominal terms using fixed-point constraints instead of freshness constraints [2]. Fixed-point constraints have the form $\pi \cdot t \overset{\curlywedge}{\approx}_\alpha t$ (read "the permutation π fixes the term t"). For nominal unification problems,

[*] Author partially funded by Capes and CNPq.

there is a direct correspondence between solutions expressed using freshness constraints and solutions expressed using fixed-point constraints, but in the presence of commutative theories the method via fixed point stands out.

Let $\approx_{\alpha,\mathtt{C}}$ denote α-equivalence modulo commutativity. Using freshness constraints, the equation $(a\ b)\cdot X \approx^?_{\alpha,\mathtt{C}} X$ has a unifier $\langle a,b\#X, Id\rangle$ [9], but this is not the only solution. Indeed there are infinite solutions $\{X \mapsto a+b\}, \{X \mapsto (a+b)+(a+b)\}, \{X \mapsto f(a+b)\},\ldots$ While nominal unification is finitary [9], when equational theories are involved, this property is lost if solutions are represented using freshness constraints, as shown in [1]. Note that $(a\ b)\cdot(a+b) = b+a \approx_{\alpha,\mathtt{C}} a+b$, so the permutation $(a\ b)$ fixes the term $a+b$, although the atoms a and b occur free in $a+b$. With the fixed-point approach, the nominal unification algorithm (modulo commutativity) computes a finite complete set of unifiers [2]. Fixed-point equations appeared for the first time in [8] in the context of nominal unification with recursive let, but the authors used the standard notion of freshness to develop their work.

Recently, in [3], a method was proposed to decide the nominal disunification problem, which is an extension of the first-order disunification problem defined in [4], where the α-equivalence relation is built using permutations and freshness. The idea is simple: one checks if the set of equations associated to the disequations, i.e., $\langle u_j \approx^?_\alpha v_j (1\leq j\leq m)\rangle$ is satisfiable, if yes (with solution set S), remove the solutions of $\langle s_i \approx^?_\alpha t_i (1\leq i\leq n)\rangle$ that are instances of S. Therefore, the proposed nominal disunification algorithm relies on the existence of a finite representation of solutions for nominal unification problems. For this reason, the nominal disunification algorithm proposed in [3], which represents solutions using freshness constraints, cannot be used to solve nominal C-disunification problems.

In this work we define the *nominal disunification problem via fixed-point constraints* (Definition 3.1) and we extend to this approach several concepts necessary for the study of its decidability: a new notion of *solution* for this problem (Definition 3.4) which depends on the concept of *pair with exceptions* (Definition 3.2) as well as its consistency (Definition 3.3). We prove consistency results (Corollary 3.1) and present the algorithm (Algorithm 2) to obtain a complete set of solutions (Theorem 3.1). This is a first step towards the development of extensions of the nominal disunification problem that involve equational theories.

2 Preliminaries

We assume familiarity with nominal techniques and briefly recall basic notions for a fixed-point approach to nominal syntax. For a detailed treatment, the reader is referred to [2].

Nominal Terms. We fix countable infinite pairwise-disjoint sets of *atoms* $\mathbb{A} = \{a,b,c,\ldots\}$, *variables* $\mathbb{X} = \{X,Y,\ldots\}$ and a signature Σ, a finite set of function symbols with fixed arity. We follow Gabbay's *permutative convention*: atoms a,b range permutatively over $\mathbb{A}$. A *permutation* π is a bijection $\mathbb{A}\to\mathbb{A}$ such that $\mathtt{dom}(\pi) := \{a\in\mathbb{A} \mid \pi(a)\neq a\}$ is finite. The identity permutation is $\mathtt{id}$ and $\pi\circ\rho$ the composition of π and ρ.

Nominal terms are given by the grammar: $s,t := a \mid \pi\cdot X \mid [a]t \mid f(t_1,\ldots,t_n)$ where a is an *atom*, $\pi\cdot X$ is a moderated variable, $[a]t$ is the *abstraction* of a in the term t, and $f(t_1,\ldots,t_n)$ is a *function application* with $f : n \in \Sigma$. We abbreviate an ordered sequence $t_1,\ldots,t_n$ of terms by $\tilde{t}$. 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 $\pi\cdot f(t_1,\ldots,t_n) = f(\pi\cdot t_1,\ldots\pi\cdot t_n)$. *Substitutions* are finite mappings from variables to terms. A substitution σ is lifted to a map over terms by: $a\sigma = a$, $(\pi\cdot X)\sigma = \pi\cdot(X\sigma)$, $([a]t)\sigma = [a](t\sigma)$, and $f(t_1,\ldots,t_n)\sigma = f(t_1\sigma,\ldots,t_n\sigma)$. Note that $t(\sigma\gamma) = (t\sigma)\gamma$.

$$\frac{\pi(a)=a}{\Upsilon\vdash\pi\curlywedge a}\,(\curlywedge\mathbf{a})\qquad \frac{\mathtt{dom}(\pi^{\rho^{-1}})\subseteq\mathtt{dom}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon\vdash\pi\curlywedge\rho\cdot X}\,(\curlywedge\mathbf{v})\qquad \frac{\Upsilon\vdash\pi\curlywedge t_i}{\Upsilon\vdash\pi\curlywedge f(\tilde{t})}\,(\curlywedge\mathbf{f})\qquad \frac{}{\Upsilon\vdash a\overset{\curlywedge}{\approx}_\alpha a}\,(\overset{\curlywedge}{\approx}_\alpha\mathbf{a})$$

$$\frac{\Upsilon,(c_1\ c_2)\curlywedge\mathtt{Var}(t)\vdash\pi\curlywedge(a\ c_1)\cdot t}{\Upsilon\vdash\pi\curlywedge[a]t}\,(\curlywedge\mathbf{ab})\qquad \frac{\Upsilon\vdash t_i\overset{\curlywedge}{\approx}_\alpha t'_i}{\Upsilon\vdash f(\tilde{t})\overset{\curlywedge}{\approx}_\alpha f(\tilde{t'})}\,(\overset{\curlywedge}{\approx}_\alpha\mathbf{f})\qquad \frac{\Upsilon\vdash t\overset{\curlywedge}{\approx}_\alpha t'}{\Upsilon\vdash[a]t\overset{\curlywedge}{\approx}_\alpha[a]t'}\,(\overset{\curlywedge}{\approx}_\alpha[\mathbf{a}])$$

$$\frac{\mathtt{dom}((\pi')^{-1}\circ\pi)\subseteq\mathtt{dom}(\mathtt{perm}(\Upsilon|_X))}{\Upsilon\vdash\pi\cdot X\overset{\curlywedge}{\approx}_\alpha\pi'\cdot X}\,(\overset{\curlywedge}{\approx}_\alpha\mathbf{v})\qquad \frac{\Upsilon\vdash s\overset{\curlywedge}{\approx}_\alpha(a\ b)\cdot t\quad \Upsilon,(c_1\ c_2)\curlywedge\mathtt{Var}(t)\vdash(a\ c_1)\curlywedge t}{\Upsilon\vdash[a]s\overset{\curlywedge}{\approx}_\alpha[b]t}\,(\overset{\curlywedge}{\approx}_\alpha\mathbf{ab})$$

Table 1: Fixed-point and equality rules. c_1 and c_2 are new atoms.

Term-equality via fixed-point constraints. Following the fixed-point approach [2], we axiomatise nominal α-equivalence in terms of both a fixed-point $\curlywedge$ and term-equality $\overset{\curlywedge}{\approx}_\alpha$ predicates. Intuitively, $\pi\curlywedge t$ means that π has no effect on t except by permuting abstracted names, while $s\overset{\curlywedge}{\approx}_\alpha t$ means that s and t are α-equivalent. For instance, $(a\ b)\curlywedge[a]a$ but not $(a\ b)\curlywedge f(a)$. In order to formally define judgement derivations for fixed-point ($\Upsilon\vdash\pi\curlywedge t$) and term-equality ($\Upsilon\vdash s\overset{\curlywedge}{\approx}_\alpha t$), we need to introduce some notation.

A fixed-point context Υ is a set of *primitive fixed-point constraints* of the form $\pi\curlywedge X$. 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. The set $\mathtt{Var}(\Upsilon)$ contains all the variables mentioned in the fixed-point context Υ and $\mathtt{perm}(\Upsilon|_X):=\{\pi\mid\pi\curlywedge X\in\Upsilon\}$ as the set of permutations of Υ with respect to $X\in\mathtt{Var}(\Upsilon)$. We write $\pi\curlywedge\mathtt{Var}(t)$ as an abbreviation for $\{\pi\curlywedge X\mid X\in\mathtt{Var}(t)\}$. Derivability for the judgements $\Upsilon\vdash\pi\curlywedge t$ and $\Upsilon\vdash s\overset{\curlywedge}{\approx}_\alpha t$ is therefore defined by the derivation rules in Figure 1.

Nominal Unification via Fixed-Point Constraints. A *nominal unification problem* $\mathtt{Pr}$ is a finite set of fixed-point ($\pi\curlywedge^? t$) and equality ($s\overset{\curlywedge}{\approx}{}^?_\alpha t$) constraints. A solution to this problem is a pair $\langle\Phi,\sigma\rangle$ consisting of a context Φ and substitution σ satisfying: (i) $\Phi\vdash\pi\curlywedge t\sigma$, if $\pi\curlywedge^? t\in\mathtt{Pr}$, and (ii) $\Phi\vdash s\sigma\overset{\curlywedge}{\approx}_\alpha t\sigma$, if $s\overset{\curlywedge}{\approx}{}^?_\alpha t\in\mathtt{Pr}$. As usual, the set of solutions $\mathcal{U}(\mathtt{Pr})$ for $\mathtt{Pr}$ is ordered via an instantiation ordering: $\langle\Phi_1,\sigma_1\rangle\leq\langle\Phi_2,\sigma_2\rangle$ iff there exists a substitution δ such that $\Phi_2\vdash X\sigma_2\overset{\curlywedge}{\approx}_\alpha X\sigma_1\delta$ and $\Phi_2\vdash\Phi_1\delta$, for all $X\in\mathbb{X}$. In this case, the pair $\langle\Phi_2,\sigma_2\rangle$ is an *instance* of the pair $\langle\Phi_1,\sigma_1\rangle$.

In [2] a rule-based algorithm ($\mathtt{unify}_\curlywedge$) was proposed to compute solutions of these problems, if any exists. It applies the rules in Table 1 bottom-up plus rules for instantiating variables:

$$\begin{array}{lll} (\overset{\curlywedge}{\approx}_\alpha\ inst1) & \mathtt{Pr}\uplus\{\pi\cdot X\overset{\curlywedge}{\approx}{}^?_\alpha t\}\overset{[X\mapsto\pi^{-1}\cdot t]}{\Longrightarrow} & \mathtt{Pr}\{X\mapsto\pi^{-1}\cdot t\},\ \text{if } X\notin\mathtt{Var}(t)\\ (\overset{\curlywedge}{\approx}_\alpha\ inst2) & \mathtt{Pr}\uplus\{t\overset{\curlywedge}{\approx}{}^?_\alpha\pi\cdot X\}\overset{[X\mapsto\pi^{-1}\cdot t]}{\Longrightarrow} & \mathtt{Pr}\{X\mapsto\pi^{-1}\cdot t\},\ \text{if } X\notin\mathtt{Var}(t)\end{array}$$

The algorithm was shown to be terminating, sound, and complete.

A matching-in-context problem has the form $(\Phi\vdash s)\approx_?(\Upsilon\vdash t)$, it is a version of the nominal unification problem in which only one side can be instantiated (here, the right-hand side) and, in addition, the contexts Φ,Υ have to be satisfied. A solution to a matching problem is a substitution δ satisfying: $\Phi\vdash s\overset{\curlywedge}{\approx}_\alpha t\delta,\Upsilon\delta$. Matching-in-context, also called pattern-matching, was introduced in [5] to define nominal rewriting (using freshness constraints).

3 Nominal Disunification via Fixed-Point Constraints

This section contains the contributions that extend [3]. Next, we define the nominal disunification problem via fixed-point constraints.

Definition 3.1. A nominal disunification problem $\mathcal{P}_\curlywedge$ is a pair $\langle \texttt{Pr} \mid\mid D_\curlywedge\rangle$ of the form $\mathcal{P}_\curlywedge = \langle\ \texttt{Pr} \mid\mid u_1 \stackrel{\curlywedge}{\not\approx}{}_\alpha^? v_1, \dots, u_m \stackrel{\curlywedge}{\not\approx}{}_\alpha^? v_m\rangle$, where $\texttt{Pr}$ is a nominal unification problem and $D_\curlywedge$ consists of a finite (possibly empty) set of nominal disequations.

In contrast with unification, where we are interested in solving equations, disunification problems enrich unification problems with disequations. The intuition is that disequations are constraints on the way we instantiate solutions of the equations we want to solve. For instance, $\mathcal{P}_\curlywedge = \langle X \stackrel{\curlywedge}{\approx}{}_\alpha^? f(Y) \mid\mid Y \stackrel{\curlywedge}{\not\approx}{}_\alpha^? a, Y \stackrel{\curlywedge}{\not\approx}{}_\alpha^? b\rangle$ expresses that, while solving the equation $X \stackrel{\curlywedge}{\approx}{}_\alpha^? f(Y)$, a solution must also satisfy the constraint that no instance of it is allowed to map X to $f(a)$ nor X to $f(b)$. Therefore, while $[X \mapsto f(c)]$, which is a grounding instance of the mgu $[X \mapsto f(Y)]$, validates the constraints imposed by $\mathcal{P}_\curlywedge$, the instance $[X \mapsto f(a)]$ does not.

The next example illustrates this principle when fixed-point constraints are taken into account.

Example 3.1. Consider $\mathcal{P}_\curlywedge = \langle X \stackrel{\curlywedge}{\approx}{}_\alpha^? (a\ b)\cdot Y \mid\mid [a]X \stackrel{\curlywedge}{\not\approx}{}_\alpha^? [b]Y\rangle$. The substitution $[X \mapsto (a\ b)\cdot Y]$ solves the equational part. In order to get the set of constraints imposed by $[a]X \stackrel{\curlywedge}{\not\approx}{}_\alpha^? [b]Y$, we solve the equation $[a]X \stackrel{\curlywedge}{\not\approx}_\alpha [b]X$ associated to it. The equation associated to the disequation is $[a]X \stackrel{\curlywedge}{\approx}{}_\alpha^? [b]Y$. Using the rule ($\stackrel{\curlywedge}{\approx}_\alpha \mathbf{ab}$) in Table 1, the fact that the system is syntax-directed, and satisfies the inversion property it follows that $[a]X \stackrel{\curlywedge}{\approx}{}_\alpha^? [b]Y$ *iff* $\{X \stackrel{\curlywedge}{\approx}{}_\alpha^? (a\ b)\cdot Y,\ (a\ c_1) \curlywedge^? Y, (c_1\ c_2) \curlywedge^? Y\}$, where c_1, c_2 are new names.

The solutions for $\mathcal{P}_\curlywedge$ are all the instances of the pair $\langle\Phi, \sigma\rangle = \langle\emptyset, [X \mapsto (a\ b)\cdot Y]\rangle$ that do not satisfy $(a\ c_1) \curlywedge^? Y, (c_1\ c_2) \curlywedge^? Y$.

The example above shows that we need the information of new names that are generated when we solve equations associated to disequations.

Definition 3.2. Let $\mathcal{P}_\curlywedge$ be a disunification problem. A *pair with exceptions for* $\mathcal{P}_\curlywedge$, denoted as $\langle\Phi, \sigma\rangle - \Theta$, consists of a pair $\langle\Phi, \sigma\rangle$ and an indexed family Θ of the form $\{\langle\nabla_l^1 \uplus \nabla_l^2, \theta_l\rangle \mid l \in I\}$, where ∇_l^2 is a (possibly empty) set of primitive fixed-point constraints involving new names, i.e., names not occurring anywhere in $\mathcal{P}_\curlywedge$.

The notion of pair with exceptions is key for the representation of solutions of a disunification problem: it will impose restrictions (the exceptions) on how these solutions can be instantiated. Intuitively, Θ consists of pairs of solutions of the equations associated to the disequations in $\mathcal{P}_\curlywedge$.

Definition 3.3. Let $\mathcal{P}_\curlywedge$ be a disunification problem. Let $\overline{c}$ be a set of new atoms and $\overline{X}$ be the set of variables of $\mathcal{P}_\curlywedge$. We denote by $\Phi^{\overline{c},\overline{X}}$ the extension of Φ with a set of primitive constraints $(c_1\ c_2) \curlywedge X$ for every pair c_1, c_2 in $\overline{c}$ and $X \in \overline{X}$. We say that

(i) $\langle\Phi, \sigma\rangle$ is an instance of a family $\Theta = \{\langle\nabla_l^1 \uplus \nabla_l^2, \theta_l\rangle \mid l \in I\}$ iff every instance of $\langle\Phi^{\overline{c},\overline{X}}, \sigma\rangle$, is an instance of some $\langle\nabla_l^1 \uplus \nabla_l^2, \theta_l\rangle \in \Theta$, where ∇_l^2 consists of primitive fixed-point constraints involving new names, $\overline{c}$ are all the new names occurring in ∇_l^2 for any l.

(ii) $\langle\Delta, \lambda\rangle$ is an instance of $\langle\Phi, \sigma\rangle - \Theta$ iff $\langle\Delta, \lambda\rangle$ is an instance of $\langle\Phi, \sigma\rangle$ but not of Θ.

(iii) A pair with exceptions $\langle \Phi, \sigma\rangle - \Theta$ is consistent iff it has at least one instance.

Lemma 3.1 (Inconsistency Lemma)**.** A pair with exceptions $\langle \Phi, \sigma\rangle - \Theta$ is inconsistent if and only if $\langle \Phi, \sigma\rangle$ is an instance of Θ.

The next result is the basis for an algorithm (Algorithm 1) to test the consistency of a pair with exceptions $\langle \Phi, \sigma\rangle - \Theta$ for $\mathcal{P}_{\curlywedge}$. It suffices to solve *matching-in-context* problems of the form $(\Phi^{\overline{c},\overline{X}} \vdash X\sigma) \approx_? (\nabla_l^1 \uplus \nabla_l^2 \vdash X\theta_l)$ for every variable X in $\mathcal{P}_{\curlywedge}$, Differently from [3], here we check the domain of the permutations in $\Phi^{\overline{c},\overline{X}}|_X$ and in an instance[1] $\langle(\nabla_l^1 \uplus \nabla_l^2)\delta\rangle_{\mathtt{nf}}|_X$ for X in $\mathcal{P}_{\curlywedge}$.

Corollary 3.1. Let $\langle \Phi, \sigma\rangle - \Theta$ be a pair with exceptions for $\mathcal{P}_{\curlywedge}$. If there is some $\langle \nabla_l^1 \uplus \nabla_l^2, \theta_l\rangle \in \Theta$ such that there exists a solution δ for the matching-in-context problems $(\Phi^{\overline{c},\overline{X}} \vdash X\sigma) \approx_? (\nabla_l^1 \uplus \nabla_l^2 \vdash X\theta_l)$, for all $X \in \mathcal{P}_{\curlywedge}$, then $\langle \Phi, \sigma\rangle - \Theta$ is inconsistent. Moreover, $\mathtt{dom}(\mathtt{perm}(\langle(\nabla_l^1 \uplus \nabla_l^2)\delta\rangle_{\mathtt{nf}}|_X)) \subseteq \mathtt{dom}(\mathtt{perm}((\Phi^{\overline{c},\overline{X}}|_X))$ for each X.

The corollary provides a method for checking for consistency of a pair with exceptions for a problem $\mathcal{P}_{\curlywedge}$:

Algorithm 1: Consistency Test

```
input: a finite pair with exceptions $\langle \Phi, \sigma\rangle - \Theta$ for $\mathcal{P}_{\curlywedge}$.
output: true if the input is consistent, false, otherwise.
for $\langle \nabla_l^1 \uplus \nabla_l^2, \theta_l\rangle \in \Theta$ do
    if $\delta = \mathtt{matching}(\Phi^{\overline{c},\overline{X}},\ X_1\sigma \approx_? X_1\theta_l, \cdots,\ X_n\sigma \approx_? X_n\theta_l)$
    then
        if $\mathtt{dom}(\mathtt{Perm}(\langle(\nabla_l^1 \uplus \nabla_l^2)\delta\rangle_{\mathtt{nf}}|_X)) \subseteq \mathtt{dom}(\mathtt{Perm}(\Phi^{\overline{c},\overline{X}}|_X))$, for all $X \in \mathcal{P}_{\curlywedge}$
        then
          return false and stop
        end if
    end if
end for
return true
```

A solution for a disunification problem $\mathcal{P}_{\curlywedge}$ will be a pair $\langle \Phi, \sigma\rangle$ that satisfies the conjunction of constraints in $E_{\curlywedge}$, and the conjunction of the constraints in $D_{\curlywedge}$. Formally,

Definition 3.4. Let $\mathcal{P}_{\curlywedge} = \langle \mathtt{Pr} \mid\mid p_1 \overset{\curlywedge\,?}{\not\approx}_\alpha q_1, \dots, p_m \overset{\curlywedge\,?}{\not\approx}_\alpha q_m\}\rangle$ be a nominal disunification problem. A *solution* to $\mathcal{P}_{\curlywedge}$ is a pair $\langle \Delta, \lambda\rangle$ of a context Δ and a substitution λ satisfying the following conditions:

1. $\langle \Delta, \lambda\rangle$ is a solution for $\mathtt{Pr}$ of $\mathcal{P}_{\curlywedge}$;

2. $\langle \Delta, \lambda\rangle$ satisfies the disequational part $D_{\curlywedge}$ of $\mathcal{P}_{\curlywedge}$, that is, for all grounding substitution δ:

$$\Delta \not\vdash (p_1\lambda\delta \overset{\curlywedge}{\approx}_\alpha q_1\lambda\delta \vee \ldots \vee p_n\lambda\delta \overset{\curlywedge}{\approx}_\alpha q_n\lambda\delta).$$

Definition 3.5. We call a set S of pairs with exceptions for $\mathcal{P}_{\curlywedge}$ a *complete representation* of the solutions of the constraint problem $\mathcal{P}_{\curlywedge}$ iff S satisfies the following conditions:

1. if $\langle \Phi, \sigma\rangle - \Theta \leq \langle \Delta, \lambda\rangle$ for some $\langle \Phi, \sigma\rangle - \Theta$ in S, then $\langle \Delta, \lambda\rangle$ solves $\mathcal{P}_{\curlywedge}$;

2. if $\langle \Delta, \lambda\rangle$ solves $\mathcal{P}_{\curlywedge}$, then it is an instance of some $\langle \Phi, \sigma\rangle - \Theta$ in S;

[1]The normal form of the instance $(\nabla_l^1 \uplus \nabla_l^2)\delta$ of the context $\nabla_l^1 \uplus \nabla_l^2$ w.r.t. the rules in $\mathtt{unify}_{\curlywedge}$

3. $\langle \Phi, \sigma \rangle - \Theta$ is consistent for all $\langle \Phi, \sigma \rangle - \Theta$ in S.

Theorem 3.1 (Representation Theorem)**.** Let $\mathcal{P}_{\curlywedge} = \langle \mathtt{Pr} \mid\mid D_{\curlywedge} \rangle$ be a nominal disunification problem. Define the family

$$\Theta := \bigcup_{p_i \overset{\curlywedge ?}{\not\approx}_\alpha q_i \in D_\curlywedge} \mathcal{U}(p_i \overset{\curlywedge ?}{\approx}_\alpha q_i).$$

Then the set $S = \{\langle \Phi, \sigma \rangle - \Theta \mid \langle \Phi, \sigma \rangle \in \mathcal{U}(\mathtt{Pr}) \text{ and } \Theta \not\lesssim \langle \Phi, \sigma \rangle\}$ is a complete representation of solutions for the problem $\mathcal{P}_\curlywedge$.

Algorithm 2: Construction of a complete representation of solutions

```
input: A problem $\mathcal{P}_\curlywedge = \langle \mathtt{Pr} \mid\mid D_\curlywedge \rangle$.
output: A finite set $S$ of pair pairs with exceptions (possibly empty).
let $\langle \Phi, \sigma \rangle := \mathtt{unify}_\curlywedge(\mathtt{Pr})$
let
$\Theta := \bigcup_{p_i \overset{\curlywedge ?}{\not\approx}_\alpha q_i \in D_\curlywedge} \{\langle \nabla_i^1 \uplus \nabla_i^2, \theta_i \rangle = \mathtt{unify}_\curlywedge(p_i \overset{\curlywedge ?}{\approx}_\alpha q_i)\}$
if consistent($\langle \Phi, \sigma \rangle - \Theta$) then
return $\langle \Phi, \sigma \rangle - \Theta$
else return $\emptyset$
end if
```

Example 3.2. Let $\mathcal{P}'_\curlywedge = \langle \underbrace{\{(a\ c_1) \curlywedge^? Y, (c_1\ c_2) \curlywedge^? Y, X \overset{\curlywedge ?}{\approx}_\alpha (a\ b) \cdot Y\}}_{\mathtt{Pr}} \mid\mid [a]X \overset{\curlywedge ?}{\not\approx}_\alpha [b]Y \rangle$. Applying Algorithm 2 one has:

- $\mathtt{unify}_\curlywedge(\mathtt{Pr}) = \langle \{(a\ c_1) \curlywedge Y, (c_1\ c_2) \curlywedge Y\}, [X \mapsto (a\ b) \cdot Y] \rangle = \langle \Phi, \sigma \rangle$, and
- $\Theta = \{\langle \underbrace{\{(a\ c_1') \curlywedge Y, (c_1'\ c_2') \curlywedge Y\}}_{\nabla}, \underbrace{[X \mapsto (a\ b) \cdot Y]}_{\theta} \rangle\} = \mathtt{unify}_\curlywedge([a]X \overset{\curlywedge ?}{\approx}_\alpha [b]Y)$ where c_1' and c_2' are new names.

Inconsistency of $\langle \Phi, \sigma \rangle - \Theta$ follows from Algorithm 1:

- $\Phi^{\overline{c}, \overline{X}} = \Phi^{c_1', c_2', X, Y} = \Phi \cup \{(c_1'\ c_2') \curlywedge X, (c_1'\ c_2') \curlywedge Y\}$.
- $id = \mathtt{matching}(\Phi^{c_1', c_2', X, Y}, X\sigma \approx_? X\theta, Y\sigma \approx_? Y\theta)$ and
- $\mathtt{dom}(\mathtt{perm}(\langle \nabla id \rangle_{\mathtt{nf}}|_Y)) = \{c_1', c_2', a\} \subseteq \mathtt{dom}(\mathtt{perm}(\Phi^{c_1', c_2', X, Y})|_Y) = \{a, c_1, c_2, c_1', c_2'\}$.
- $\mathtt{dom}(\mathtt{perm}(\langle \nabla id \rangle_{\mathtt{nf}}|_X)) = \emptyset \subseteq \mathtt{dom}(\mathtt{perm}(\Phi^{c_1', c_2', X, Y})|_X) = \{c_1', c_2'\}$.

Therefore, $S = \emptyset$ and there is no solution for $\mathcal{P}_\curlywedge$.

4 Conclusion and Future Work

This work used the fixed-point relation, intrinsic to the definition of the freshness relation, in order to extend the syntax concepts already defined in the usual nominal disunification. The fixed-point approach proved to be useful for dealing with equational theories that involve commutativity. For this reason, in future work, we intend to finalise the semantic analysis of our extension and take advantage of its finite representation of solutions to investigate disunification problems involving equational theories.

References

[1] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, and Daniele Nantes-Sobrinho. On solving nominal fixpoint equations. In *Front. of Combining Systems - 11th Int. Symp., FroCoS 2017, Proc.*, volume 10483 of *LNCS*, pages 209–226. Springer, 2017. doi:10.1007/978-3-319-66167-4_12.

[2] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. On nominal syntax and permutation fixed points. *LMCS*, 16(1), 2020. doi:10.23638/LMCS-16(1:19)2020.

[3] Mauricio Ayala-Rincón, Maribel Fernández, Daniele Nantes-Sobrinho, and Deivid Vale. On solving nominal disunification constraints. In *Proc. LSFA 2019*. doi:10.1016/j.entcs.2020.02.002.

[4] Wray L. Buntine and Hans-Jürgen Bürckert. On solving equations and disequations. *J. ACM*, 41(4):591–629, 1994. doi:10.1145/179812.179813.

[5] Maribel Fernández and Murdoch Gabbay. Nominal rewriting. *Inf. Comput.*, 205(6):917–965, 2007. doi:10.1016/j.ic.2006.12.002.

[6] Murdoch Gabbay and Andrew M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects Comput.*, 13(3-5):341–363, 2002. doi:10.1007/s001650200016.

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

[8] Manfred Schmidt-Schauß, Temur Kutsia, Jordi Levy, and Mateu Villaret. Nominal unification of higher order expressions with recursive let. In Manuel V. Hermenegildo and Pedro López-García, editors, *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR 2016, Edinburgh, UK, September 6-8, 2016, Revised Selected Papers*, volume 10184 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2016. doi:10.1007/978-3-319-63139-4_19.

[9] Christian Urban, Andrew M. Pitts, and Murdoch Gabbay. Nominal unification. *Theor. Comput. Sci.*, 323(1-3):473–497, 2004. doi:10.1016/j.tcs.2004.06.016.