

# Closed nominal rewriting and efficiently computable nominal algebra equality

Maribel Fernández and Murdoch J. Gabbay

**Abstract.** We analyse the relationship between nominal algebra and nominal rewriting, giving a new and concise presentation of equational deduction in nominal theories. With some new results, we characterise a subclass of equational theories for which nominal rewriting provides a complete and efficient procedure to check nominal algebra equality. This subclass includes specifications of lambda-calculus and first-order logic.

## 1 Introduction

It is very common, when formally defining a programming language, computation model, or deduction system, to appeal to operators with binding like  $\forall$ ,  $\lambda$ ,  $\nu$ , or  $\int$ . We are therefore interested in frameworks with support for the specification, analysis and evaluation of operators with binding mechanisms.

Two such frameworks are nominal algebra [GM06,GM09a] and nominal rewriting [FGM04,FG07]. Both are theories of equality. Both are based on nominal terms [UPG04,FG07], which extend first-order syntax with binding while retaining a first-order flavour (e.g. unification is decidable). Operators with binding can be represented and their properties axiomatised using equational rules or rewrite rules. The representation is natural and closely follows informal practice. Example ‘nominal’ theories are in this paper (Examples 4 and 17) and in [UPG04,FG07,GM09a]. The reader can also find in [GM08b,GM08b,GM08a] more extended treatments of nominal theories for first-order logic, lambda-calculus, and substitution, respectively, along with proofs of soundness and completeness.

The relationship between equational deduction (roughly, the “replacement of equals by equals”) and rewriting is well-understood in the first-order case, where terms do not include binders. In this case, if an equational theory  $E$  can be presented by a terminating and confluent set of rewrite rules, then equality modulo  $E$  is decidable [DJ89,BN98]. Even if the rewrite rules are not confluent it may be possible to use rewriting if the system can be completed by adding new rules, as suggested by Knuth and Bendix [KB70]. Several implementations of equational logic have been built on this basis (see, for instance, [BM79,O’D87] and [GSH<sup>+</sup>92,McC97,McC03]).

In systems with binding the situation is different. Semi-automatic tools exist, many relying on higher-order formalisms that use the  $\lambda$ -calculus as meta-language. However, since higher-order matching is undecidable, higher-order rewriting frameworks need to restrict the form of the rules to achieve a decidable rewriting relation. Nominal techniques provide an alternative approach to the specification of equality for systems with binding, but the relationship

between nominal rewriting and equational theorem proving is not straightforward, and has not yet been established.

One goal of this paper is to fill this gap, and to show that, for a large class of theories of interest, nominal rewriting can serve as a basis for the implementation of equational reasoning. For this, there are some issues that we need to address. Firstly, there is no clear presentation in the literature of nominal algebra and nominal rewriting side-by-side. Secondly, equivariant nominal unification is decidable, but it is also computationally very expensive [Che04]. Unfortunately equivariant nominal matching is the underlying computational ‘engine’ for nominal rewriting and nominal algebra. Thirdly and most importantly, nominal algebra and nominal rewriting appear to implement subtly different notions of equality on nominal terms; if we orient nominal algebra equalities and take the reflexive transitive symmetric closure, we get back a weaker notion of equality than we started with. In this paper do the following:

- We give new presentations of nominal rewriting and nominal algebra that are significantly more concise than those in [FG07,GM09a]. This gives a clear and ‘user-friendly’ overview of the two systems.
- As it turns out, a naive identification of nominal rewriting with “oriented equality” in nominal algebra, fails. If we ‘just orient’ equalities to obtain nominal rewrites, in the style of moving from universal algebra to first-order rewriting, then we obtain a weaker theory than expected, even if the rewrite rules are confluent (Lemma 22).

We can, however, identify an indirect completeness result (Theorem 23). This makes a new and precise completeness connection between nominal rewriting and nominal algebra.

- Nominal rewriting is not efficiently computable, but *closed* nominal rewriting is. We present closed nominal rewriting and show that for closed rules, closed rewriting is sound and complete for nominal algebra (Theorem 46), and in a direct manner.

The idea of closed nominal rewriting and closed rules, and the fact that they are efficiently computable, was already known in [FGM04] and [FG07]. Closed nominal rewriting and closed rules are also expressive; at least as expressive as other systems for rewriting-with-binding in the literature, including Combinatory Reduction Systems [KvOvR93] and Higher-Order Rewriting Systems [MN98].

What is new to this paper is that we put closed nominal rewriting in the context of nominal algebra equality and prove that it directly induces exactly the right notion of equality for nominal algebra.

Summarising, we will show how nominal rewriting corresponds to nominal algebra, and also we will show that closed rewriting provides a direct and computationally tractable mechanism to implement nominal algebra.

The rest of the paper is organised as follows: In Section 2 we recall the basic notions of nominal syntax. Section 3 gives a new and uniform presentation of nominal algebra and nominal rewriting. Section 4 compares nominal algebra and rewriting and establishes a first completeness result. Section 5 discusses

closed nominal rewriting as an efficient mechanism to implement deduction in nominal theories, and contains the main result of the paper: the soundness and completeness of nominal rewriting for equational deduction in theories presented by closed rules. Section 6 describes an algorithm to implement nominal algebra in an efficient way. We conclude the paper in Section 7.

## 2 Preliminaries

### 2.1 Terms and signatures

**Definition 1.** Fix disjoint countably infinite collections of **atoms**, **unknowns** (or variables), and **term-formers**. We use a **permutative convention** that  $a, b, c, \dots$  range over distinct atoms; we write  $\mathbb{A}$  for the set of all atoms.  $X, Y, Z, \dots$  will range over distinct unknowns.  $f, g, \dots$  will range over distinct term-formers. We assume that to each term-former  $f$  is associated some unique **arity**  $n$  which is a nonnegative number; we write  $f : n$  to indicate that  $f$  has arity  $n$ . A **signature**  $\Sigma$  is a set of term-formers with their arities.

**Definition 2.** A **permutation**  $\pi$  is a bijection on atoms such that  $\text{atoms}(\pi) = \{a \mid \pi(a) \neq a\}$  is finite.

Write  $(a \ b)$  for the **swapping** permutation that maps  $a$  to  $b$ ,  $b$  to  $a$  and all other  $c$  to themselves, and  $\text{id}$  for the **identity permutation**, so  $\text{id}(a) = a$  always. Write  $\pi \circ \pi'$  for **functional composition** of permutations, so  $(\pi \circ \pi')(a) = \pi(\pi'(a))$ , and  $\pi^{-1}$  for **inverse**, so  $\pi(a) = b$  if and only if  $a = \pi^{-1}(b)$ .

**Definition 3. (Nominal) terms**  $t, u, v$  are inductively defined by:

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

We write  $\equiv$  for syntactic identity, so  $t \equiv u$  when  $t$  and  $u$  denote the same term.

Call  $[a]t$  an **(atoms-)abstraction**; it represents  $'x.e'$  or  $x.\phi'$  in expressions like  $'\lambda x.e'$  or  $'\forall x.\phi'$ . We define an  $\alpha$ -equivalence relation  $\approx_\alpha$  later, in Definition 9.

**Example 4.** Here are some example nominal terms:

– Suppose term-formers  $\text{lam} : 1$  and  $\text{app} : 2$ . Then:

$$\begin{array}{ll} \text{lam}([a]\text{lam}([b]\text{app}(a, b))) & \text{represents the } \lambda\text{-term } \lambda f.\lambda x.fx \\ \text{lam}([a]\text{lam}([b]X)) & \text{represents a } \lambda\text{-term schema } \lambda x.\lambda y.t \end{array}$$

– Suppose term-formers  $\perp : 0$ ,  $\supset : 2$ ,  $\forall : 1$ , and  $\approx : 2$ . Then:

$$\begin{array}{ll} \forall[a](\forall[b](a \approx b)) & \text{represents the sentence } \forall x.\forall y.(x = y) \\ \forall[a]X & \text{represents the sentence schema } \forall x.\phi \end{array}$$

### 2.2 Permutation and substitution

**Definition 5.** Define an **(object-level) permutation action**  $\pi \cdot t$  by:

$$\begin{array}{ll} \pi \cdot a \equiv \pi(a) & \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \\ \pi \cdot [a]t \equiv [\pi(a)](\pi \cdot t) & \pi \cdot f(t_1, \dots, t_n) \equiv f(\pi \cdot t_1, \dots, \pi \cdot t_n) \end{array}$$

$$\begin{array}{c}
\frac{}{\Delta \vdash a \# b} (\# \mathbf{ab}) \qquad \frac{}{\Delta \vdash a \# [a]t} (\#[\mathbf{a}]) \\
\frac{(\pi^{-1}(a) \# X) \in \Delta}{\Delta \vdash a \# \pi \cdot X} (\#\mathbf{X}) \qquad \frac{\Delta \vdash a \# t}{\Delta \vdash a \# [b]t} (\#[\mathbf{b}]) \\
\frac{\Delta \vdash a \# t_1 \cdots \Delta \vdash a \# t_n}{\Delta \vdash a \# f(t_1, \dots, t_n)} (\#f) \\
\frac{\Delta \vdash b \# t \quad \Delta \vdash (b a) \cdot t \approx_\alpha u}{\Delta \vdash [a]t \approx_\alpha [b]u} (\approx_\alpha \mathbf{[b]}) \quad \frac{(a \# X \in \Delta \text{ for all } a \text{ s.t. } \pi(a) \neq \pi'(a))}{\Delta \vdash \pi \cdot X \approx_\alpha \pi' \cdot X} (\approx_\alpha \mathbf{X}) \\
\frac{\Delta \vdash t \approx_\alpha u}{\Delta \vdash [a]t \approx_\alpha [a]u} (\approx_\alpha \mathbf{[a]}) \quad \frac{\Delta \vdash t_i \approx_\alpha u_i \quad (1 \leq i \leq n)}{\Delta \vdash f(t_1, \dots, t_n) \approx_\alpha f(u_1, \dots, u_n)} (\approx_\alpha f)
\end{array}$$

Fig. 1: Freshness and  $\alpha$ -equality

A **substitution (on unknowns)**  $\sigma$  is a partial function from unknowns to terms with finite domain.  $\theta$  and  $\sigma$  will range over substitutions.

Define a **(meta-level) substitution action**  $t\sigma$  by:

$$\begin{array}{l}
a\sigma \equiv a \qquad (\pi \cdot X)\sigma \equiv \pi \cdot X \quad (X \notin \text{dom}(\sigma)) \\
([a]t)\sigma \equiv [a](t\sigma) \qquad (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) \quad (X \in \text{dom}(\sigma)) \\
f(t_1, \dots, t_n)\sigma \equiv f(t_1\sigma, \dots, t_n\sigma)
\end{array}$$

Henceforth, if  $X \notin \text{dom}(\sigma)$  then  $\sigma(X)$  denotes  $\text{id} \cdot X$ .

Write  $\text{id}$  for the substitution with  $\text{dom}(\text{id}) = \emptyset$ , so that  $t\text{id} \equiv t$  always. When we write  $\text{id}$ , it will always be clear whether we mean ‘ $\text{id}$  the identity substitution’ or ‘ $\text{id}$  the identity permutation’ (Definition 2).

If  $\sigma$  and  $\theta$  are substitutions, write  $\sigma \circ \theta$  for the function mapping  $X$  to  $(X\sigma)\theta$ .

Lemmas 6, 7 and 8 are proved by routine inductions:

**Lemma 6.**  $(\pi \circ \pi') \cdot t \equiv \pi \cdot (\pi' \cdot t)$  and  $\text{id} \cdot t \equiv t$ .

**Lemma 7.**  $\pi \cdot (t\sigma) \equiv (\pi \cdot t)\sigma$ .

**Lemma 8.**  $(r\sigma)\theta \equiv r(\sigma \circ \theta)$ .

### 2.3 $\alpha$ -equivalence

The native notion of equality on nominal terms is  $\alpha$ -equivalence (for comparison, that of first-order terms is syntactic identity, and that of higher-order terms is  $\beta$ -equivalence or possibly  $\beta\eta$ -equivalence). The advantage of this notion of equality is that it gives us a notion of binding, but retains a first-order spirit.

**Definition 9.** A **freshness (constraint)** is a pair  $a\#t$  of an atom  $a$  and a term  $t$ . Call a freshness of the form  $a\#X$  **primitive**, and call a finite set of primitive freshneses a **freshness context**.  $\Delta, \Gamma$  and  $\nabla$  will range over freshness contexts.

We may drop set brackets and write  $a\#t, b\#u$  for  $\{a\#t, b\#u\}$ . Also, we may write  $a\#t, u$  for  $a\#t, a\#u$ , and  $a, b\#t$  for  $a\#t, b\#t$ .

A **freshness judgement** is a tuple  $\Delta \vdash a\#t$  of a freshness context and a freshness constraint. An  **$\alpha$ -equivalence judgement** is a tuple  $\Delta \vdash s \approx_\alpha t$  of a freshness context and two terms. Define the **derivable** freshness and  $\alpha$ -equivalence judgements by the rules in Figure 1.

**Definition 10.** Define  $atoms(t)$  and  $unknowns(t)$  by:

$$\begin{aligned} atoms(a) &= \{a\} & atoms(\pi \cdot X) &= atoms(\pi) \\ atoms([a]t) &= atoms(t) \cup \{a\} & atoms(f(t_1, \dots, t_n)) &= \bigcup_i atoms(t_i) \\ unknowns(a) &= \emptyset & unknowns(\pi \cdot X) &= \{X\} \\ unknowns([a]t) &= unknowns(t) & unknowns(f(t_1, \dots, t_n)) &= \bigcup_i unknowns(t_i) \end{aligned}$$

**Definition 11.** Later in this paper, starting with Definition 29, we find it useful to write  $atoms(X)$  and  $unknowns(X)$  for  $X$  something more complex than a term — e.g. a list (as in ‘ $atoms(\Delta, s, t)$ ’), a term-in-context (as in ‘ $unknowns(\nabla \vdash l)$ ’), or a substitution. By this we mean the atoms or unknowns appearing anywhere within the brackets. So  $atoms(\Delta, s, t)$  means  $\{a \mid a\#X \in \Delta \text{ for some } X\} \cup atoms(s) \cup atoms(t)$ . Also,  $atoms(\theta) = \bigcup \{atoms(\theta(X)) \mid X \in dom(\theta)\}$ .

**Lemma 12 (Strengthening).** Suppose  $a \notin atoms(s, t)$ . Then:

- $\Delta, a\#X \vdash b\#s$  implies  $\Delta \vdash b\#s$ .
- $\Delta, a\#X \vdash s \approx_\alpha t$  implies  $\Delta \vdash s \approx_\alpha t$ .

*Proof.* By induction on the rules in Figure 1, using the fact that in all cases the hypotheses of rules use only atoms already mentioned in the conclusions.

**Definition 13.** Suppose  $S$  is a set of freshness constraints and  $\theta$  is a substitution. Define  $S\theta = \{a\#(s\theta) \mid a\#s \in S\}$ .

**Lemma 14 (Weakening).** Suppose  $\Delta \vdash \Delta' \sigma$ . Then

- $\Delta' \vdash b\#s$  implies  $\Delta \vdash b\#s\sigma$  always.
- $\Delta' \vdash s \approx_\alpha t$  implies  $\Delta \vdash s\sigma \approx_\alpha t\sigma$  always.

In particular, taking  $\sigma = id$  and  $\Delta' = \Delta, \Gamma$ , we obtain:

- $\Delta \vdash b\#s$  implies  $\Delta, \Gamma \vdash b\#s$  always.
- $\Delta \vdash s \approx_\alpha t$  implies  $\Delta, \Gamma \vdash s \approx_\alpha t$  always.

*Proof.* By routine inductions on the rules in Figure 1.

### 3 Nominal algebra and nominal rewriting

We define notions of equational theory and rewriting over nominal terms. Because nominal terms have a native notion of binding, theories inherit this and can exploit it to axiomatise properties of binding operators (e.g. it is direct and natural to axiomatise  $\beta$ -equivalence [GM08c]).

**Remark 15.** The notions of equality and rewriting in this paper correspond to those introduced in [GM09a] and [FG07], respectively. However, the presentations of nominal rewriting and nominal algebra in (1) and (2) of Definition 19 below, are original to this paper. They are not quite the same as the original presentations of [FG07] (nominal rewriting) and [GM09a] (nominal algebra). Arguably, Definition 19 contains the clearest presentation of nominal rewriting and nominal algebra so far. It is certainly the most concise.

We have also optimised the presentations to make it particularly easy to compare and contrast the two notions — to bring out what they have in common, and what is different.

Just a little checking needs to be done, to verify that (1) and (2) *really do* define nominal rewriting and nominal algebra as presented in the literature. This is not very hard, but it is not obvious either. For completeness we provide a sketch of the argument in Appendix A.

**Definition 16.** We introduce two new judgement-forms:

- An **equality judgement** is a tuple  $\Delta \vdash s = t$  of a freshness context and two terms.
- A **rewrite judgement** is a tuple  $\Delta \vdash s \rightarrow t$  of a freshness context and two terms.

We may write ‘ $\emptyset \vdash$ ’ as ‘ $\vdash$ ’.

We also introduce two notions of theory — one for equality judgements, and one for rewrite judgements:

- An **equational theory**  $T = (\Sigma, Ax)$  is a pair of a signature  $\Sigma$  and a possibly infinite set of equality judgements  $Ax$  in that signature; we call them **axioms**.
- A **rewrite theory**  $R = (\Sigma, Rw)$  is a pair of a signature  $\Sigma$  and a possibly infinite set of rewrite judgements  $Rw$  in that signature; we call these **rewrite rules**.

We may omit  $\Sigma$ , identifying  $T$  with  $Ax$  and  $R$  with  $Rw$  when the signature is clear from the context.

**Example 17.** Assume term-formers  $\lambda : 1$ ,  $\text{app} : 2$ , and  $\text{sub} : 2$ . Write  $\text{app}(s, t)$  as  $st$  and write  $\text{sub}([a]s, t)$  as  $s[a \mapsto t]$ . Rules for  $\beta$ -reduction are:

$$\begin{array}{lll}
 (\text{Beta}) & (\lambda[a]X).X' & \rightarrow X[a \mapsto X'] \\
 (\sigma_{\text{app}}) & (XX')[a \mapsto Y] & \rightarrow X[a \mapsto Y]X'[a \mapsto Y] \\
 (\sigma_{\text{var}}) & a[a \mapsto X] & \rightarrow X \\
 (\sigma_{\epsilon}) & a\#Y \vdash Y[a \mapsto X] & \rightarrow Y \\
 (\sigma_{\text{lam}}) & b\#Y \vdash (\lambda[b]X)[a \mapsto Y] & \rightarrow \lambda[b](X[a \mapsto Y])
 \end{array}$$

Other rules are also possible, e.g.  $a\#X \vdash X \rightarrow \lambda[a](Xa)$  for  $\eta$ -expansion.

Of course, we obtain a nominal algebra theory just by replacing  $\rightarrow$  with  $=$ .

**Definition 18.** A **position**  $C$  is a pair  $(s, X)$  of a term and a distinguished unknown  $X$  that occurs precisely once in the syntax of  $s$ , as  $id \cdot X$ .

If  $C = (s, X)$  then write  $C[t]$  for  $s[X \mapsto t]$ .

We are now ready to define notions of derivable equality, and rewriting:

**Definition 19.** Write  $\Delta \vdash (\phi_1, \dots, \phi_n)$  for the judgements  $\Delta \vdash \phi_1, \dots, \Delta \vdash \phi_n$ .

- **Nominal rewriting**  $\Delta \vdash_{\mathbf{R}} s \rightarrow t$  is the least transitive reflexive relation such that for every  $(\nabla \vdash l \rightarrow r) \in \mathbf{R}$ , freshness context  $\Delta$ , position  $C$ , term  $s'$ , permutation  $\pi$ , and substitution  $\theta$ ,

$$\frac{s \equiv C[s'] \quad \Delta \vdash (\nabla \theta, \quad s' \approx_{\alpha} \pi \cdot (l\theta), \quad C[\pi \cdot (r\theta)] \approx_{\alpha} t)}{\Delta \vdash_{\mathbf{R}} s \rightarrow t} \text{ (Rew}_{\nabla \vdash l \rightarrow r}\text{)}. \quad (1)$$

- **(Nominal algebra) equality**  $\Delta \vdash_{\mathbf{T}} s = t$  is the least transitive reflexive symmetric relation such that for every  $(\nabla \vdash l = r) \in \mathbf{T}$  and freshness context  $\Delta$ , position  $C$ , permutation  $\pi$ , substitution  $\theta$ , and fresh  $\Gamma$  (so if  $a \# X \in \Gamma$  then  $a \notin \text{atoms}(\Delta, s, t)$ ),

$$\frac{\Delta, \Gamma \vdash (\nabla \theta, \quad s \approx_{\alpha} C[\pi \cdot (l\theta)], \quad C[\pi \cdot (r\theta)] \approx_{\alpha} t)}{\Delta \vdash_{\mathbf{T}} s = t} \text{ (Axi}_{\nabla \vdash l = r}\text{)}. \quad (2)$$

## 4 Soundness and completeness of nominal rewriting with respect to nominal algebra

Proposition 21 and Theorem 23 describe how nominal rewriting from (1) is sound and complete for nominal algebra from (2).

**Definition 20.** Suppose  $\mathbf{T}$  is a nominal algebra theory and  $\mathbf{R}$  is a nominal rewrite theory. Call  $\mathbf{R}$  a **presentation** of  $\mathbf{T}$  when

$$\nabla \vdash s = t \in \mathbf{T} \quad \Leftrightarrow \quad (\nabla \vdash s \rightarrow t \in \mathbf{R} \quad \vee \quad \nabla \vdash t \rightarrow s \in \mathbf{R}).$$

Write  $\Delta \vdash_{\mathbf{R}} s \leftrightarrow t$  for the symmetric closure  $\Delta \vdash_{\mathbf{R}} s \rightarrow t$ .

**Proposition 21** (Soundness). *Suppose  $\mathbf{R}$  is a presentation of  $\mathbf{T}$ .*

*Then  $\Delta \vdash_{\mathbf{R}} s \leftrightarrow t$  implies  $\Delta \vdash_{\mathbf{T}} s = t$ .*

*Proof.* By a routine induction on the derivation  $\Delta \vdash_{\mathbf{R}} s \leftrightarrow t$ . We briefly sketch the case of  $(\text{Rew}_{\nabla \vdash l \rightarrow r})$  for  $\nabla \vdash l = r \in \mathbf{T}$ .

Suppose for some  $C, \theta$ , and  $\pi$ ,

$$s \equiv C[s'] \quad \text{and} \quad \Delta \vdash (\nabla \theta, \quad s' \approx_{\alpha} \pi \cdot (l\theta), \quad C[\pi \cdot (r\theta)] \approx_{\alpha} t).$$

Let  $\Gamma = \emptyset$ . It is a fact that if  $\Delta \vdash s' \approx_{\alpha} \pi \cdot (l\theta)$  then  $\Delta \vdash C[s'] \approx_{\alpha} C[\pi \cdot (l\theta)]$ . We now easily obtain an instance of  $(\text{Axi}_{\nabla \vdash l = r})$ .

**Lemma 22.** *Suppose  $\mathbf{R}$  is a presentation of  $\mathbf{T}$ . It is not necessarily the case that  $\Delta \vdash_{\mathbf{T}} s = t$  implies  $\Delta \vdash_{\mathbf{R}} s \leftrightarrow t$ .*

*Proof.*  $R = \{(\vdash [b]b \rightarrow b)\}$  is a presentation of  $T = \{(\vdash [b]b = b)\}$ . It is easy to verify that  $\vdash_{\tau} [b][a]a = [a]b$  but  $\not\vdash_r [b][a]a \leftrightarrow [a]b$ .

**Theorem 23** (Completeness). *Suppose  $R$  is a presentation of  $T$ .*

*Then  $\Delta \vdash_{\tau} s = t$  implies that there exists some fresh  $\Gamma$  (so if  $a \# X \in \Gamma$  then  $a \notin \text{atoms}(\Delta, s, t)$ ) such that  $\Delta, \Gamma \vdash_r s \leftrightarrow t$ .*

Note the ‘fresh  $\Gamma$ ’ on the side of nominal rewriting.

*Proof.* We work by induction on the derivation of  $\Delta \vdash_{\tau} s = t$ , write it  $\Pi$ .

The interesting case is  $(\mathbf{Axi}_{\nabla \vdash l = r})$  for some  $\nabla \vdash l = r \in T$ , of course. There is only one argument in the proof that is not entirely obvious:  $\Pi$  is finite, so let us consider *all* the finitely many instances of  $(\mathbf{Axi})$  in  $\Pi$ ; write them  $I_1, \dots, I_n$ . For each  $1 \leq i \leq n$ ,  $I_i$  will involve  $\nabla_i \vdash l_i = r_i$ ,  $C_i$ ,  $\pi_i$ ,  $\theta_i$ , and a fresh context  $\Gamma_i$ . (Note that  $\Delta$  is constant across all these instances.)

Now it is a fact that the atoms in  $\Gamma_i$  do not feature in the conclusion of  $I_i$  because the atoms in  $\Gamma_i$  do not feature in  $\Delta$ ,  $C_i$ ,  $\pi_i$ , and  $\theta_i$ .

Therefore, it is a fact that we can rename these atoms so that they are fresh for all parts of  $\Pi$  other than hypotheses of instances of  $(\mathbf{Axi})$ . That is; there exists a derivation  $\Pi'$  of  $\Delta \vdash_{\tau} s = t$  such that for each  $1 \leq i \leq n$  the respective  $\Gamma'_i$  in respective instances  $I'_i$  of  $(\mathbf{Axi})$  are fresh not only locally for the conclusion of  $I'_i$ , but also fresh globally for all conclusions of all  $I'_j$  for  $1 \leq j \leq n$  in  $\Pi'$ .

This ‘global freshness’ condition is clearly preserved by taking subderivations. We take  $\Gamma = \bigcup_i \Gamma'_i$ . The proof is now by a routine induction on  $\Pi'$ .

**Remark 24.** The proof of Theorem 23 yields an upper bound on the size of  $\Gamma$ : the maximal size of the  $\Gamma_i$ . Note that the construction does not require the  $\Gamma_i$  to be renamed to be fresh for each other; they need only be renamed to be fresh for all the rest of the derivation.

## 5 Closed rewriting and nominal algebra

### 5.1 The definition of closed rules and closed rewriting

**Definition 25** (Terms-in-context and nominal matching). A **term-in-context** is a pair  $\Delta \vdash s$  of a freshness context and a term. A **nominal matching problem** is a pair of terms-in-context

$$(\nabla \vdash l) \text{ ?} \approx (\Delta \vdash s) \quad \text{where } \text{unknowns}(\nabla \vdash l) \cap \text{unknowns}(\Delta \vdash s) = \emptyset.$$

A **solution** to this problem is a substitution  $\sigma$  such that

$$\Delta \vdash \nabla \sigma \quad \text{and} \quad \Delta \vdash l \sigma \approx_{\alpha} s \quad \text{and} \quad \text{dom}(\sigma) \subseteq \text{unknowns}(\nabla \vdash l).$$

**Remark 26.** Nominal matching is decidable [UPG04], and can be solved in linear time [CF08].

**Definition 27** (Freshened variants). If  $t$  is a term, call  $t'$  a **freshened variant** of  $t$  when  $t'$  has the same structure as  $t$ , except that the atoms and unknowns have been replaced by ‘fresh’ atoms and unknowns. We omit an inductive definition.

Similarly, if  $\nabla$  is a freshness context then  $\nabla'$  will denote a freshened variant of  $\nabla$  (so if  $a\#X \in \nabla$  then  $a'\#X' \in \nabla'$ , where  $a'$  and  $X'$  are chosen fresh).

We may extend this to other syntax, like equality and rewrite judgements.

Note that if  $\nabla' \vdash l' \rightarrow r'$  is a freshened variant of  $\nabla \vdash l \rightarrow r$  then  $unknowns(\nabla' \vdash l' \rightarrow r') \cap unknowns(\nabla \vdash l \rightarrow r) = \emptyset$ .

**Example 28.** For example:

–  $[a'][b']X'$  is a freshened variant of  $[a][b]X$ ,  $a'\#X'$  is a freshened variant of  $a\#X$ , and  $\emptyset \vdash a' \rightarrow b'$  is a freshened variant of  $\emptyset \vdash a \rightarrow b$ .

– Neither  $[a'][a']X'$  nor  $[a'][b']X$  are freshened variants of  $[a][b]X$  (the first, because we have wrongly identified two distinct atoms when we freshened them; the second, because we did not freshen  $X$ ).

**Definition 29.** Call a term-in-context  $\nabla \vdash l$  **closed** when the matching problem

$$(\nabla' \vdash l') \text{ ?} \approx (\nabla, atoms(l')\#unknowns(\nabla, l) \vdash l) \quad (3)$$

has a solution  $\sigma$ . Here,  $\nabla', l'$  are freshened with respect to  $\nabla$  and  $l$ .

**Remark 30.** It does not matter which freshened variant of  $\nabla \vdash l$  we choose in Definition 29, so long as it is fresh. Similarly for the definition of *closed rewriting* in Definition 32.

This fact is closely related to the some/any property of the  $\mathbb{U}$ -quantifier [GP01], and to the principle of ZFA equivariance described e.g. in [GM09a, Theorem A.4]. Intuitively, a valid way to look at Definitions 29 and 32 is that the atoms in  $\nabla' \vdash l'$  are ‘bound’, or ‘occupy a separate namespace’.

**Remark 31.** We unpack what Definitions 29 and 25 mean.  $\nabla \vdash l$  is closed when there exists a substitution  $\sigma$  with  $dom(\sigma) \subseteq unknowns(\nabla \vdash l)$  such that

$$\nabla, atoms(l)\#unknowns(\nabla, l) \vdash (\nabla'\sigma, l \approx_\alpha l'\sigma).$$

**Definition 32.** – Call  $R = (\nabla \vdash l \rightarrow r)$  and  $A = (\nabla \vdash l = r)$  **closed** when  $\nabla \vdash (l, r)$  is closed.

– Given a rewrite rule  $R = (\nabla \vdash l \rightarrow r)$  and a term-in-context  $\Delta \vdash s$ , write  $\Delta \vdash s \xrightarrow{R}_c t$  when there is some  $R'$  a freshened variant of  $R$  (so fresh for  $R, \Delta, s$ , and  $t$ ), position  $C$  and substitution  $\theta$  such that

$$s \equiv C[s'] \quad \text{and} \quad \Delta, atoms(R')\#unknowns(\Delta, s, t) \vdash (\nabla'\theta, s' \approx_\alpha l'\theta, C[r'\theta] \approx_\alpha t). \quad (4)$$

We call this **closed rewriting**.

**Remark 33.** Comparing Definition 32 (closed rewriting) with Definition 19 (rewriting) we see they are very similar. However, there are two key differences:

– The  $\pi$  in (1) in Definition 19 is not there in (4) in Definition 32. This greatly increases the efficiency of calculating closed nominal rewrites.

- Atoms cannot ‘interact by name’ in a closed rewrite step. For instance, suppose  $R = (a \rightarrow b)$ . Then  $\vdash a \xrightarrow{R} b$  and  $\vdash c \xrightarrow{R} d$ . However,  $\not\vdash a \xrightarrow{R}_c b$  and  $\not\vdash c \xrightarrow{R}_c d$ . Atoms can still take part in closed nominal rewriting. For instance, if  $R = ([a]X \rightarrow [a]X)$  then  $\vdash [a]a \xrightarrow{R} [a]a$  and also  $\vdash [a]a \xrightarrow{R}_c [a]a$ .

**Example 34.** The rules in Example 17 are closed.

$fa \rightarrow a$  is not a closed rule, neither are  $fa \rightarrow b$ , or  $[a]X \rightarrow X$ , but  $a\#X \vdash [a]X \rightarrow X$  is closed.

## 5.2 Properties of closed rewriting, and connection with nominal algebra

**Definition 35.** Define  $\sigma \circ \pi$  by:

$$\begin{aligned} (\sigma \circ \pi)(X) &= \pi \cdot (\sigma(X)) && \text{if } X \in \text{dom}(\sigma) \\ (\sigma \circ \pi)(X) & \text{undefined} && \text{otherwise.} \end{aligned}$$

**Lemma 36.** If  $\text{atoms}(s) \cap \text{atoms}(\pi) = \emptyset$  then  $\pi \cdot (s\sigma) \equiv s(\sigma \circ \pi)$ .

*Proof.* By a routine induction on  $s$ .

**Definition 37.** Write  $\Delta \vdash \sigma \approx_\alpha \sigma'$  when  $\text{dom}(\sigma) = \text{dom}(\sigma')$  and  $\Delta \vdash \sigma(X) \approx_\alpha \sigma'(X)$  for every  $X \in \text{dom}(\sigma)$ .

- Lemma 38.**
1. Suppose  $a \notin \text{atoms}(s', l')$ . Then if  $\Delta \vdash s' \approx_\alpha l'\sigma$  then there exists some  $\sigma'$  such that  $\Delta \vdash \sigma \approx_\alpha \sigma'$  and  $a \notin \text{atoms}(\sigma')$ .
  2. Suppose  $a \notin \text{atoms}(t, r', C)$ . Then if  $\Delta \vdash C[r'\sigma] \approx_\alpha t$  then there exists some  $\sigma'$  such that  $\Delta \vdash \sigma \approx_\alpha \sigma'$  and  $a \notin \text{atoms}(\sigma')$ .

*Proof.* For the first part, we construct  $\sigma'$  by an induction on the structure of  $l'$ . We sketch one case:

- The case  $l' \equiv \pi \cdot X$ . By assumption  $\Delta \vdash s' \approx_\alpha \pi \cdot \sigma(X)$ , where  $a \notin \text{atoms}(\pi)$ . We choose  $\sigma'(X) \equiv \pi^{-1}(s')$ .

For the second part we work by induction on the derivation of  $\Delta \vdash C[r'\sigma] \approx_\alpha t$ , using the rules in Figure 1 to break down the syntax of  $C$  until we reach the first case (note that  $\approx_\alpha$  is symmetric).

**Proposition 39** (Strengthening for closed rewriting). Fix a context  $\Delta$  and terms  $s$  and  $t$ . Suppose  $\Gamma$  is fresh (so if  $a\#X \in \Gamma$  then  $a \notin \text{atoms}(s, t, \Delta)$ ). Suppose  $R = (\nabla \vdash l \rightarrow r)$  is a rewrite rule.

Then  $\Delta, \Gamma \vdash s \xrightarrow{R}_c t$  if and only if  $\Delta \vdash s \xrightarrow{R}_c t$ .

*Proof.* Suppose  $\Delta, \Gamma \vdash s \xrightarrow{R}_c t$ . Unpacking definitions, this means that there is some  $R'$  a freshened version of  $R$  (with respect to  $s, t, \Delta, \Gamma$ , and  $R$ ), and some position  $C$  and substitution  $\sigma$  such that  $\text{dom}(\sigma) \subseteq \text{unknowns}(R')$  and

$$s \equiv C[s'] \quad \Delta, \Gamma, \text{atoms}(R')\#\text{unknowns}(R) \vdash (\nabla' \sigma, s' \approx_\alpha l'\sigma, C[r'\sigma] \approx_\alpha t).$$

Using Lemma 38 we may assume without loss of generality that  $a \notin \text{atoms}(\sigma)$ . By elementary calculations on the atoms of terms and using Strengthening (Lemma 12) we deduce

$$s \equiv C[s'] \quad \Delta, \text{atoms}(R') \# \text{unknowns}(R) \vdash (\nabla' \sigma, \quad s' \approx_\alpha l' \sigma, \quad C[r' \sigma] \approx_\alpha t).$$

That is,  $\Delta \vdash s \xrightarrow{R}_c t$  as required.

Conversely, suppose  $\Delta \vdash s \xrightarrow{R}_c u$ . We unpack definitions as before and use Weakening (Lemma 14).

**Lemma 40.**  $\Delta \vdash a \# s$  if and only if  $\Delta \vdash \pi(a) \# \pi \cdot s$ .

*Proof.* By induction on the freshness derivation rules in Figure 1.

Proposition 41 below is part of [FG07, Theorem 70]. Although the result is known, in our opinion the proof we now give is an improvement (clearer, and more self-contained):

**Proposition 41.** If  $R = (\nabla \vdash l \rightarrow r)$  is closed then  $\Delta \vdash_R s \rightarrow t$  implies  $\Delta \vdash_R s \rightarrow_c t$ .

*Proof.* Suppose  $\Delta \vdash_R s \rightarrow t$ . So there exist  $\Delta, C, s', \pi$ , and  $\theta$  such that

$$s \equiv C[s'] \quad \text{and} \quad \Delta \vdash (\nabla \theta, \quad s' \approx_\alpha \pi \cdot (l\theta), \quad C[\pi \cdot (r\theta)] \approx_\alpha t).$$

$\nabla \vdash l \rightarrow r$  is closed so by Remark 31 there exists a freshened variant  $R' = (\nabla' \vdash l' \rightarrow r')$  of  $R$  and a substitution  $\sigma$  such that  $\text{dom}(\sigma) \subseteq \text{unknowns}(R')$  and

$$\nabla, \text{atoms}(l') \# \text{unknowns}(\nabla, l) \vdash (\nabla' \sigma, \quad l \approx_\alpha l' \sigma, \quad r \approx_\alpha r' \sigma).$$

It follows using Lemmas 14 and 7 that

$$s \equiv C[s'] \quad \Delta, \text{atoms}(l') \# \text{unknowns}(\nabla, l) \vdash (\nabla' \sigma \theta, \quad s' \approx_\alpha \pi \cdot (l' \sigma \theta), \quad C[\pi \cdot (r' \sigma \theta)] \approx_\alpha t).$$

By assumption the atoms in  $R'$  are fresh and so we can assume  $\text{atoms}(R') \cap \text{atoms}(\pi) = \emptyset$ . It follows by Lemmas 36 and 8 that  $\pi \cdot (l' \sigma \theta) \equiv l'((\sigma \circ \theta) \circ \pi)$  and  $\pi \cdot (r' \sigma \theta) \equiv r'((\sigma \circ \theta) \circ \pi)$ . Using Lemma 40  $\Delta, \text{atoms}(l') \# \text{unknowns}(\nabla, l) \vdash \nabla'((\sigma \circ \theta) \circ \pi)$  also follows. Write  $\theta'$  for  $(\sigma \circ \theta) \circ \pi$ . Then

$$s \equiv C[s'] \quad \Delta, \text{atoms}(l') \# \text{unknowns}(\nabla, l) \vdash (\nabla' \theta', \quad s' \approx_\alpha l' \theta', \quad C[r' \theta'] \approx_\alpha t).$$

That is,  $\Delta \vdash s \xrightarrow{R}_c t$  as required.

**Definition 42.** Suppose  $\nabla \vdash l$  is a term-in-context. Suppose  $\text{atoms}(\nabla \vdash l) = \{a_1, \dots, a_n\}$  and  $\text{unknowns}(\nabla \vdash l) = \{X_1, \dots, X_n\}$  (where we take these atoms and unknowns in some fixed but arbitrary order).

Suppose  $\nabla' \vdash l'$  is a freshened variant of  $\nabla \vdash l$  and  $\text{atoms}(\nabla' \vdash l') = \{a'_1, \dots, a'_n\}$  and  $\text{unknowns}(\nabla' \vdash l') = \{X'_1, \dots, X'_n\}$  (where we take these fresh atoms and unknowns in a corresponding order).

Define a permutation  $\tau$  and substitution  $\varsigma$  by

$$\tau(a'_1 a_1) \circ \dots \circ (a'_n a_n) \quad \text{and} \quad \varsigma[X'_1 \mapsto \tau \cdot X_1, \dots, X'_n \mapsto \tau \cdot X_n].$$

**Lemma 43.** *Suppose  $\nabla \vdash l$  is a closed term-in-context. Suppose  $\nabla' \vdash l'$  is a freshened variant of  $\nabla \vdash l$ . Define  $\tau$  and  $\varsigma$  as in Definition 42. Then:*

1.  $l' \equiv \tau \cdot (l\varsigma)$ .
2.  $\Gamma' \vdash \nabla'\theta$  if and only if  $\Gamma \vdash \nabla\varsigma\theta$ .

*Proof.* We prove the first part by induction on  $l$ . We sketch the case of  $\pi \cdot X$ :

$$\tau \cdot ((\pi \cdot X)\varsigma) \stackrel{\text{Lemma 6}}{\equiv} (\tau \circ \pi) \cdot \varsigma(X) \stackrel{\text{Lemma 6}}{\equiv} (\tau \circ \pi \circ \tau) \cdot X' \stackrel{\text{fact}}{\equiv} \pi' \cdot X'.$$

For the second part consider some  $a' \# X' \in \nabla'$  (originating from  $a \# X \in \nabla$ ). By definition  $\varsigma(X) \equiv \tau \cdot X'$  and it follows that

$$X'\theta \stackrel{\text{Lemma 6}}{\equiv} (\tau \cdot (X\varsigma))\theta \stackrel{\text{Lemma 7}}{\equiv} \tau \cdot (X\varsigma\theta).$$

By Lemma 40  $\Gamma' \vdash a' \# (X'\theta)$  if and only if  $\Gamma \vdash a \# (X\varsigma\theta)$ . The result follows.

**Proposition 44.** *If  $R = (\nabla \vdash l \rightarrow r)$  is closed then  $\Delta \vdash_R s \rightarrow_c t$  implies there is some fresh  $\Gamma$  (so if  $a \# X \in \Gamma$  then  $a \notin \text{atoms}(\Delta, s, t)$ ) such that  $\Delta, \Gamma \vdash s \xrightarrow{R} t$ .*

*Proof.* If  $\Delta \vdash_R s \rightarrow_c t$  then for some freshened variant  $R' = (\nabla' \vdash l' \rightarrow r')$  of  $R$  (freshened with respect to  $R, \Delta, s$ , and  $t$ ) there exists some position  $C$ , term  $s'$ , and substitution  $\theta$  such that

$$s \equiv C[s'] \quad \Delta, \text{atoms}(R') \# \text{unknowns}(\Delta, s, t) \vdash (\nabla'\theta, \quad s' \approx_\alpha l'\theta, \quad C[r'\theta] \approx_\alpha t).$$

By Lemmas 43 and 7

$$s \equiv C[s'] \quad \Delta, \text{atoms}(R') \# \text{unknowns}(\Delta, s, t) \vdash (\nabla\varsigma\theta, \quad s' \approx_\alpha \tau \cdot (l\varsigma\theta), \quad C[\tau \cdot (r'\varsigma\theta)] \approx_\alpha t).$$

Using Lemmas 8 and 7 we deduce  $\Delta, \text{atoms}(R') \# \text{unknowns}(\Delta, s, t) \vdash s \xrightarrow{R} t$ .

**Definition 45.** Write  $\Delta \vdash_R s \leftrightarrow_c t$  for the symmetric closure of  $\Delta \vdash s \xrightarrow{R} t$ .

**Theorem 46** (Soundness and completeness). *Suppose the rewrite theory  $R$  is a presentation (Definition 20) of the equational theory  $T$ . Suppose all rules in  $R$  are closed.*

*Then  $\Delta \vdash_T s = t$  if and only if  $\Delta \vdash_R s \leftrightarrow_c t$ .*

*Proof.* By Proposition 21 and Theorem 23  $\Delta \vdash_T s = t$  if and only if  $\Delta, \Gamma \vdash_R s \leftrightarrow t$  for some fresh  $\Gamma$ . By Proposition 44  $\Delta, \Gamma \vdash_R s \leftrightarrow t$  for some fresh  $\Gamma$  if and only if  $\Delta \vdash_R s \leftrightarrow_c t$ .

## 6 Mechanising equational reasoning

We show that closed nominal rewriting can be used to automate reasoning in nominal equational theories; provided that the theory satisfies certain conditions, given an equality judgement  $\Delta \vdash s = t$  it suffices to compare the normal forms of  $s$  and  $t$  in the corresponding closed nominal rewrite system.

**Definition 47.** Call a rewrite theory  $R$  **closed** when every  $R \in R$  is closed (Definition 32).

Suppose  $R$  is a closed rewrite theory. Call a term  $s$  an **(R-)normal form**, or an **irreducible term**, when no closed nominal rewriting step can be performed (so there is no  $t$  such that  $\Delta \vdash s \xrightarrow{R}_c t$ ).

If  $\Delta \vdash s \xrightarrow{R}_c t$  and  $t$  is a normal form, call  $s$  **(R-)normalisable** and call  $t$  an **(R-)normal form** of  $s$ .

Write  $\Delta \vdash_R s \downarrow^c t$  when  $s$  closed-rewrites to  $t$  and  $t$  is a normal form.

Call  $R$  **terminating** if there are no infinite closed rewriting sequences  $\Delta \vdash t_1 \xrightarrow{R}_c t_2, t_2 \xrightarrow{R}_c t_3, \dots$

Call  $R$  **confluent** when, if  $\Delta \vdash s \xrightarrow{R}_c t$  and  $\Delta \vdash s \xrightarrow{R}_c t'$ , then  $u$  exists such that  $\Delta \vdash t \xrightarrow{R}_c u$  and  $\Delta \vdash t' \xrightarrow{R}_c u$ .

Finally, call  $R$  **convergent** when it is terminating and confluent.

Theorem 48 states that if  $T$  can be oriented to obtain a set of closed rewrite rules  $R$ , and  $R$  is confluent, then derivability in  $T$  corresponds to joinability:

**Theorem 48.** *Assume  $T$  is an equational theory such that its axioms can be oriented to form a set of closed rewrite rules  $R$ . If  $R$  is confluent, then  $\Delta \vdash_T s = t$  if and only if there exists  $u$  such that  $\Delta \vdash s \xrightarrow{R}_c u$  and  $\Delta \vdash t \xrightarrow{R}_c u$ .*

*Proof.* Consequence of Theorem 46 and confluence.

Theorem 48 does not require termination. However, termination guarantees that we can decide whether there exists a term  $u$  with the desired property: if the rewrite theory is terminating and confluent, we can rewrite  $s$  and  $t$  to normal form and then check that the normal forms are  $\alpha$ -equivalent.  $\alpha$ -equivalence is decidable, so:

**Corollary 49** (Decidability of deduction in  $T$ ). *Assume  $T$  is an equational theory such that its axioms can be oriented to form a closed set of rewrite rules  $R$ . If  $R$  is convergent, then equality is decidable in  $T$  (i.e.,  $\Delta \vdash_T s = t$  is a decidable relation). Moreover, the validity of an equational judgement can be efficiently checked: it suffices to check whether the normal forms of  $s$  and  $t$  are  $\alpha$ -equivalent.*

## 7 Conclusions

Efficient algorithms for closed nominal rewriting and for checking  $\alpha$ -equivalence are described in [CF09]. Checking that rules are closed can also be done efficiently (in linear time) with the nominal matching algorithm of [CF09]. It follows from Corollary 49 that, had we a procedure to check that a given set of rules is convergent, we could directly built an automated theorem prover for nominal theories. Unfortunately, termination and confluence are undecidable properties even for first order rules. Fortunately, closed nominal rewrite rules inherit many of the good properties of first-order rewriting systems: orthogonality is a sufficient condition for confluence (see [FGM04]) and it is very easy to check. If the theory under consideration is not orthogonal, then the alternative

is to check termination and to check that all critical pairs are joinable (which is a sufficient condition for convergence, see [FGM04]). Reduction orderings (to check termination) and completion procedures (to ensure that all critical pairs are joinable) are available for closed nominal rules [FR10].

We can consider a recent ‘permissive’ variant of nominal terms [DGM09,GM09b]. These eliminate freshness contexts and give a tighter treatment of  $\alpha$ -equivalence, which might simplify the mathematics here. Permissive nominal terms have been implemented in prototype form [Mul09], but it remains to verify that *efficient* algorithms exist to manipulate them.

## References

- BM79. Robert S. Boyer and J Strother Moore. *A Computational Logic*. Academic Press, New York, 1979.
- BN98. Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, Great Britain, 1998.
- CF08. Christophe Calvès and Maribel Fernández. Nominal matching and alpha-equivalence. In *Proceedings of WOLLIC 2008, Edinburgh, July 2008*, Lecture Notes in Artificial Intelligence. Springer, 2008.
- CF09. Christophe Calvès and Maribel Fernández. Matching and alpha-equivalence check for nominal terms. *Journal of Computer and System Sciences*, 2009. Special issue: Selected papers from WOLLIC 2008.
- Che04. James Cheney. The complexity of equivariant unification. In *Automata, Languages and Programming, Proceedings of the 31st Int. Colloquium, ICALP 2004*, volume 3142 of Lecture Notes in Computer Science. Springer, 2004.
- DGM09. Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *CILC, 24th Italian Conference on Computational Logic*, 2009.
- DJ89. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Methods and Semantics*, volume B. North-Holland, 1989.
- FG07. Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, 2007.
- FGM04. Maribel Fernández, Murdoch J. Gabbay, and Ian Mackie. Nominal Rewriting Systems. In *Proc. 6th Int. ACM SIGPLAN Conf. on Principles and Practice of Declarative Programming (PPDP’2004)*, pages 108–119. ACM Press, 2004.
- FR10. Maribel Fernández and Albert Rubio. Reduction orderings and completion of rewrite systems with binding, 2010. Available from [www.dcs.kcl.ac.uk/staff/maribel](http://www.dcs.kcl.ac.uk/staff/maribel).
- GM06. Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. In *18th Nordic Workshop on Programming Theory*, 2006.
- GM08a. Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.
- GM08b. Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.
- GM08c. Murdoch J. Gabbay and Aad Mathijssen. Reasoning in simple type theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday, chapter The lambda-calculus is nominal algebraic. *Studies in Logic and the Foundations of Mathematics*. IFCoLog, December 2008.
- GM09a. Murdoch J. Gabbay and Aad Mathijssen. Nominal universal algebra: equational logic with names and binding. *Journal of Logic and Computation*, 2009. Advance access.
- GM09b. Murdoch J. Gabbay and Dominic P. Mulligan. Universal algebra over lambda-terms and nominal terms: the connection in logic between nominal techniques and higher-order variables. In *LFMTP’09: Proceedings of the Fourth International Workshop on Logical Frameworks and Meta-Languages*, pages 64–73. ACM, 2009.
- GP01. Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.
- GSH<sup>+</sup>92. Joseph Goguen, Andrew Stevens, Hendrik Hilberdink, Keith Hobbey, W. A. Hunt, and T. F. Melham. 2obj: A metalogical framework theorem prover based on equational

- KB70. D. Knuth and P. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*. Pergamon Press, Oxford, 1970.
- KvOvR93. Jan-Willem Klop, Vincent van Oostrom, and Femke van Raamsdonk. Combinatory reduction systems, introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.
- McC97. William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19:263–276, 1997.
- McC03. William McCune. Otter 3.3 reference manual, 2003. Technical Memorandum No. 263, Argonne National Laboratory.
- MN98. Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
- Mul09. Dominic P. Mulligan. Implementation of permissive nominal terms. Available at <http://www2.macs.hw.ac.uk/~dpm8/permissive/perm.htm>, 2009.
- O'D87. Michael J. O'Donnell. Term-rewriting implementation of equational logic programming. In *Rewriting Techniques and Applications*, number 256 in Lecture Notes in Computer Science, pages 108–119. Springer, 1987.
- UPG04. Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.

## A Correspondence of the definitions of this paper with previous definitions in the literature

As mentioned in Remark 15 we should briefly consider how (1) and (2) of Definition 19 correspond to the definitions in [FG07] (nominal rewriting) and [GM09a] (nominal algebra). All the main issues are clearly indicated in the following two short sketches:

### A.1 Nominal rewriting

(1) corresponds to Definition 47 in Subsection 5.2 of [FG07]. The correspondence is clear except that Definition 47 does not include a  $\pi$ . This is because in [FG07] we insisted that rewrite theories (Definition 16 in this paper) have the additional property that they be *equivariant* (Definition 4.2 of [FG07]); that if  $R \in \mathcal{R}$  then  $R^\pi \in \mathcal{R}$  ( $R^\pi$  is  $R$  with  $\pi$  applied to all atoms). It is not hard to use Lemma 41 and part (3) of Theorem 50 in [FG07] to demonstrate that equivariance has the same effect as the  $\pi$  in (1), and indeed, if  $\Delta \vdash_{\mathcal{R}} s \rightarrow t$  then  $\Delta \vdash_{\mathcal{R}} \pi \cdot s \rightarrow \pi \cdot t$ .

In [FG07] we consider a one-step rewrite relation  $\rightarrow$ . In this paper we take as primitive the reflexive transitive closure, because this is more convenient for comparing with equality. By abuse of notation we have also written this  $\rightarrow$ .

### A.2 Nominal algebra

(2) corresponds to Definition 3.10 and to the rules in Figures 1 and 2 in [GM09a]. Figure 2 of [GM09a] has an extra rule (fr), which generates a fresh atom. This corresponds to the fresh context  $\Gamma$  in (2). Note that in (2) the fresh atoms are generated ‘all at once’ whereas in Figure 2 of [GM09a] fresh atoms may be generated at any point during equality reasoning. However, using Theorem 3.22 of [GM09a] and a routine induction it can be proved that if  $\Delta \vdash_{\mathcal{T}} r = s$  is derivable using the rules in Figures 1 and 2 of [GM09a], then there is a derivation in which all instances of (fr) occur at the head of the derivation. The interested reader can also consult Lemma 5.10 in [GM08b], where a very similar result is stated and proved in full detail, of a more complex system.