Closed nominal rewriting and efficiently computable nominal algebra equality

Maribel Fernández
www.dcs.kcl.ac.uk/staff/maribel

Murdoch J. Gabbay
www.gabbay.org.uk

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 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 $\forall$, λ, ν, or $\int$. We are therefore interested in frameworks with support for the specification, analysis and evaluation of operators with binding mechanisms. Such frameworks are needed not only in logic and theoretical computer science (where binders like $\forall$, λ and ν are familiar) but also to mechanise mathematics, linguistics, systems biology, and so on.

First, we need to define the notion of a binder. One answer is to identify all binding with functional abstraction λ. This approach is taken in the definition of higher-order abstract syntax [PE88, DPS97], higher-order rewriting [MN98], and combinatory reduction systems [KvOvR93], amongst others. Since higher-order unification is undecidable, and it plays a key role in rewriting (e.g., rewrite steps are computed using matching, critical pairs are computed using unification), most higher-order rewrite formalisms only use *higher-order patterns* [Mil91], a decidable sublanguage. This fact already suggests that names and binding might be a simpler concept, and computationally more tractable, than raw functional abstraction.

In fact, it has been shown that higher-order patterns correspond almost exactly to *nominal terms* [LV08, DGM10]. This correspondence is robust, and extends to solutions of unification problems [DGM10], and also in the presence of arbitrary equality theories [GM09b]. Unification and matching of nominal terms are decidable [UPG04] and efficient (see [CF09] for a linear-time nominal matching algorithm, and [Cal10, LV10] for efficient unification algorithms). Nominal terms have been the basis of rewriting [FG07], logic programming [CU08], and algebra [GM09a].

Nominal terms are like first-order terms ('standard' syntax) but come supplied with *atoms*; a kind of bindable constant with semantics discussed in [GP01]. Atoms display special behaviour which will be developed in the body of the paper. For now, we illustrate the use of nominal terms to express a theory of $\beta\eta$-equivalence in nominal algebra.

LFMTP 2010
Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.

© Fernández and Gabbay
This work is licensed under the
Creative Commons Attribution License.

Suppose term-formers $\mathsf{lam} : 1$ (the number indicates arity) and $\mathsf{app} : 2$. Then:

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

Here a and b are atoms, $[a]$- is *atoms-abstraction*, with special properties we discuss later, and X is a variable which corresponds to *meta-variables* like t above. But X is not a meta-variable; it is a variable in nominal terms. To avoid confusion, we call variables in nominal terms *unknowns*.

We can define β-reduction and η-reduction as follows (see [FGM04] for an alternative nominal rewriting system that uses an explicit substitution operator):

$$\begin{array}{lll} (\beta_{\mathtt{app}}) & & \mathsf{app}(\mathsf{lam}([a]\mathsf{app}(X,X')),Y) \rightarrow \\ & & \quad \mathsf{app}(\mathsf{app}(\mathsf{lam}([a]X'),Y),\ \mathsf{app}(\mathsf{lam}([a]X),Y))) \\ (\beta_{\mathtt{var}}) & & \mathsf{app}(\mathsf{lam}([a]a),X) \rightarrow X \\ (\beta_{\varepsilon}) & a\#Y \vdash & \mathsf{app}(\mathsf{lam}([a]Y),X) \rightarrow Y \\ (\beta_{\mathtt{lam}}) & b\#Y \vdash & \mathsf{app}(\mathsf{lam}([a]\mathsf{lam}([b]X)),Y) \rightarrow \mathsf{lam}([b]\mathsf{app}(\mathsf{lam}([a]X),Y)) \\ (\eta) & a\#X \vdash & \mathsf{lam}([a]\mathsf{app}(X,a)) \rightarrow X \end{array}$$

We obtain a nominal algebra theory just by replacing $\rightarrow$ with $=$.

Setting aside the verbosity of the syntax above, what we would like the reader to take from this example is *how close* the specification is to what we write in mathematical discourse. A *freshness side-condition* $a\#X$ formalises the usual condition $x \notin fv(u)$, using an atom a for x and an unknown X for the metavariable u.

This motivates nominal algebra [GM06, GM09a] and also nominal rewriting [FGM04, FG07]; theories of equality and rewriting respectively for nominal terms (see also [CP07], though this does not use nominal terms). The resulting theories have semantics in (nominal) sets and good computational properties; these are investigated in several other papers by the authors and others.

The relationship between equational reasoning and rewriting is well understood in the first-order case where terms do not include binders: If an equational theory E can be presented by a terminating and confluent rewrite system then equality modulo E is decidable [DJ89, BN98]. Even if the rewrite system is not confluent it may still be possible to use rewriting if the system can be completed by adding new rules [KB70]; implementations of equational logic have been based on these observations [BM79, O'D87, GSH+92, McC97, McC03].

However, in systems with binding the situation is different. Semi-automatic tools exist, many relying on higher-order formalisms that use the λ-calculus as meta-language, but since higher-order unification is undecidable in general, higher-order rewriting frameworks need to restrict the form of the rules to achieve a decidable rewriting relation. This makes it difficult to define completion procedures for higher-order rewriting systems. For nominal systems, the relationship between rewriting and equality is not straightforward and has not been established yet. This paper fills this gap.

The main contributions of this paper are:

- 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.
- We identify a completeness result (Theorem 4.4) which shows a precise connection between nominal rewriting and nominal algebra. In other words, we fill the gap mentioned above.

- We identify a stronger completeness result for a subset of nominal rewriting already investigated for its good computational properties [FG07]: *closed* rewriting. Closed rewriting is sound and complete for nominal algebra (Theorem 5.19), in a particularly direct manner.

Note that the collection of closed nominal terms is at least as expressive as other systems in the literature, including Combinatory Reduction Systems [KvOvR93] and Higher-Order Rewriting Systems [MN98]. This is discussed in [FG07]. However, nominal rewrite/algebra systems exist that do *not* fall into the closed collection. For instance, the natural specification of π-calculus (**Open**) labelled transition [MPW92] displays a *gensym*-like behaviour that, as it happens, is not captured by closed nominal terms (but can be defined using nominal rewriting rules): $P \xrightarrow{a\bar{b}} Q$ implies $\nu b.P \xrightarrow{a\bar{b}} Q$.[1]

So both our completeness results are relevant. We cannot say one is 'right' and the other 'wrong'; nominal terms are more expressive but fewer things are true of them relative to closed nominal terms. Both have good theorems relating rewriting with equational reasoning, which we describe in this paper.

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 establishes the soundess and completeness of nominal rewriting for equational deduction in theories presented by closed rules. Using this result, we give an algorithm to implement nominal algebra in an efficient way. We conclude the paper in Section 6.

2 Syntax and α-equivalence

Nominal terms were introduced in [UPG04] as a formal syntax for the specification of systems with binding. In this section we recall the main notions of nominal syntax; for more details and examples we refer the reader to [UPG04, FG07].

2.1 Terms and signatures

Definition 2.1. Fix disjoint countably infinite collections of **atoms**, **unknowns** (or variables), and **term-formers** (or function symbols). We write $\mathbb{A}$ for the set of atoms; $a, b, c, \ldots$ will range over distinct atoms. $X, Y, Z, \ldots$ will range over distinct unknowns. $\mathsf{f}, \mathsf{g}, \ldots$ will range over distinct term-formers. We assume that to each f is associated an **arity** n which is a nonnegative number; we write $\mathsf{f} : n$ to indicate that f has arity n. A **signature** Σ is a set of term-formers with their arities.

Definition 2.2. A **permutation** π is a bijection on atoms such that $nontriv(\pi) = \{a \mid \pi(a) \neq a\}$ is finite.

[1] This rule can be fit into the nominal algebra/rewriting framework, e.g. with a bit of sugar as follows: $(Z,\ P \xrightarrow{a\bar{b}} Q) \rightarrow (Z,\ P \xrightarrow{a\bar{b}} Q,\ \nu b.P \xrightarrow{a\bar{b}} Q)$. We are interested in expressivity, not elegance, at this point.

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

Definition 2.3. (Nominal) terms are inductively defined by:

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

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

A term of the form $[a]t$ is called an **(atom-)abstraction**; it represents '$x.e$' or '$x.\phi$' in expressions like '$\lambda x.e$' or '$\forall x.\phi$'. We define an α-equivalence relation $\approx_\alpha$ later, in Definition 2.8.

2.2 Permutation and substitution

Definition 2.4. An **(atoms) permutation action** $\pi \cdot t$ is defined by:

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

A **substitution (on unknowns)** σ is a partial function from unknowns to terms with finite domain. θ and σ will range over substitutions.

An **(unknowns) substitution action** $t\sigma$ is defined by:

$$\begin{array}{lll} a\sigma \equiv a & (\pi \cdot X)\sigma \equiv \pi \cdot X & (X \notin dom(\sigma)) \\ ([a]t)\sigma \equiv [a](t\sigma) & (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) & (X \in dom(\sigma)) \\ \mathsf{f}(t_1,\ldots,t_n)\sigma \equiv \mathsf{f}(t_1\sigma,\ldots,t_n\sigma) & & \end{array}$$

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

We write id for the substitution with $dom(id) = \varnothing$, so that $t\,id \equiv t$. When we write id, it will be clear whether we mean 'id the identity substitution' or 'id the identity permutation' (Definition 2.2).

If σ and θ are substitutions, $\sigma \circ \theta$ maps each X to $(X\sigma)\theta$.

Lemmas 2.5, 2.6 and 2.7 are proved by routine inductions (see [FG07]).

Lemma 2.5. $(\pi \circ \pi') \cdot t \equiv \pi \cdot (\pi' \cdot t)$ *and* $id \cdot t \equiv t$.

Lemma 2.6. $\pi \cdot (t\sigma) \equiv (\pi \cdot t)\sigma$.

Lemma 2.7. $t(\sigma \circ \theta) \equiv (t\sigma)\theta$.

2.3 α-equivalence

The native notion of equality on nominal terms is α-equivalence. For comparison, that of first-order terms is syntactic identity, and that of higher-order terms is β- or possibly $\beta\eta$-equivalence.

Definition 2.8. A **freshness (constraint)** is a pair $a\#t$ of an atom a and a term t. We call a freshness of the form $a\#X$ **primitive**, and a finite set of primitive freshnesses a **freshness context**. Δ, Γ and ∇ 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 $a\#t, b\#t$.

$$\frac{}{\Delta \vdash a\#b}\ (\#\mathbf{ab}) \qquad \frac{}{\Delta \vdash a\#[a]t}\ (\#[\mathbf{a}])$$

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

$$\frac{\Delta \vdash a\#t_1 \cdots \Delta \vdash a\#t_n}{\Delta \vdash a\#\mathsf{f}(t_1,\ldots,t_n)}\ (\#\mathsf{f}) \qquad \frac{}{\Delta \vdash a \approx_\alpha a}\ (\approx_\alpha \mathbf{a})$$

$$\frac{\Delta \vdash b\#t \quad \Delta \vdash (b\ a)\cdot t \approx_\alpha u}{\Delta \vdash [a]t \approx_\alpha [b]u}\ (\approx_\alpha[\mathbf{b}]) \qquad \frac{(a\#X \in \Delta \text{ for all } a \text{ s.t. } \pi(a) \neq \pi'(a))}{\Delta \vdash \pi\cdot X \approx_\alpha \pi'\cdot X}\ (\approx_\alpha \mathbf{X})$$

$$\frac{\Delta \vdash t \approx_\alpha u}{\Delta \vdash [a]t \approx_\alpha [a]u}\ (\approx_\alpha[\mathbf{a}]) \qquad \frac{\Delta \vdash t_i \approx_\alpha u_i \quad (1 \leq i \leq n)}{\Delta \vdash \mathsf{f}(t_1,\ldots,t_n) \approx_\alpha \mathsf{f}(u_1,\ldots,u_n)}\ (\approx_\alpha \mathsf{f})$$

Figure 1: Freshness and α-equality

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

Definition 2.9. The functions $atms(t)$ and $unkn(t)$ will be used to compute the set of atoms and unknowns in a term, respectively. They are defined by:

$$\begin{array}{rlrl} atms(a) &= \{a\} & atms(\pi\cdot X) &= nontriv(\pi) \\ atms([a]t) &= atms(t)\cup\{a\} & atms(\mathsf{f}(t_1,\ldots,t_n)) &= \bigcup_i atms(t_i) \\ \\ unkn(a) &= \varnothing & unkn(\pi\cdot X) &= \{X\} \\ unkn([a]t) &= unkn(t) & unkn(\mathsf{f}(t_1,\ldots,t_n)) &= \bigcup_i unkn(t_i) \end{array}$$

Definition 2.10. Later in this paper, starting with Definition 5.5, we find it useful to write $atms(\mathtt{X})$ and $unkn(\mathtt{X})$ for $\mathtt{X}$ something more complex than a term — e.g. a list (as in '$atms(\Delta,s,t)$'), a term-in-context (as in '$unkn(\nabla \vdash l)$'), or a substitution. By this we mean the atoms or unknowns appearing anywhere within the brackets. So $atms(\Delta,s,t)$ means $\{a \mid a\#X \in \Delta \text{ for some } X\} \cup atms(s)\cup atms(t)$. Also, $atms(\theta) = \bigcup\{atms(\theta(X)) \mid X \in dom(\theta)\}$.

Lemma 2.11 (Strengthening)**.** *Suppose $a \notin atms(s,t)$. 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 2.12. Suppose S is a set of freshness constraints and θ is a substitution. Define $S\theta = \{a\#(s\theta) \mid a\#s \in S\}$.

Lemma 2.13 (Weakening)**.** *Suppose $\Delta \vdash \Delta'\sigma$. Then*

- $\Delta' \vdash b\#s$ *implies* $\Delta \vdash b\#s\sigma$.

- $\Delta' \vdash s \approx_\alpha t$ *implies* $\Delta \vdash s\sigma \approx_\alpha t\sigma$.

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

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

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

3 Nominal algebra and nominal rewriting

In this section we define notions of equational reasoning and rewriting over nominal terms. Nominal terms have a native notion of binding, which theories inherit and can exploit to axiomatise properties of binding operators (e.g. it is direct and natural to axiomatise β-equivalence [GM08b]).

Definition 3.1. 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 $\Delta \vdash s \to t$ of a freshness context and two terms.

We may write '$\varnothing \vdash$' as '$\vdash$'.

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

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

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

Example 3.2. The rewrite rules $(\beta_{\mathtt{app}})$, $(\beta_{\mathtt{var}})$, (β_ε), $(\beta_{\mathtt{lam}})$, and (η) define the rewrite theory $\beta\eta$ for β- and η-reduction in the λ-calculus.

Note the use of a freshness context in rule $(\beta_{\mathtt{lam}})$ to ensure that free λ-calculus variables are not captured. In rule (β_ε) we use a freshness context to discard the argument when it is not needed. In the η rule, the freshness context formalises the usual condition on the bound variable. See [FG07] for more examples of nominal rewrite rules.

If we replace $\to$ by $=$ we obtain an equational theory. More examples of nominal equational theories can be found in [GM09a].

Definition 3.3. A **position** C is a pair (s, X) of a term and a distinguished unknown X that occurs precisely once in s, as $id \cdot X$. If $C = (s, X)$ then we write $C[t]$ for $s[X \mapsto t]$.

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

Definition 3.4. Below we write $\Delta \vdash (\phi_1, \ldots, \phi_n)$ for the judgements $\Delta \vdash \phi_1, \ldots, \Delta \vdash \phi_n$.

- **Nominal rewriting:** The **one-step rewrite relation** $\Delta \vdash s \xrightarrow{R} t$ is the least relation such that for every $(\nabla \vdash l \to r) \in \mathsf{R}$, freshness context Δ, position C, term s', permutation π, and substitution θ,

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

The **rewrite relation** $\Delta \vdash_{\mathsf{R}} s \to t$ is the reflexive transitive closure of the one-step rewrite relation, that is, the least relation that includes the one-step rewrite relation and such that:
  - for all Δ and s: $\Delta \vdash_{\mathsf{R}} s \to s'$ if $\Delta \vdash s \approx_\alpha s'$ (the native notion of equality of nominal terms is α-equality);
  - for all Δ, s, t, u: $\Delta \vdash_{\mathsf{R}} s \to t$ and $\Delta \vdash_{\mathsf{R}} t \to u$ implies $\Delta \vdash_{\mathsf{R}} s \to u$.

  If $\Delta \vdash_{\mathsf{R}} s \to t$ holds, we say that s rewrites to t in the context Δ.
- **(Nominal algebra) equality:** $\Delta \vdash_{\mathsf{T}} s = t$ is the least transitive reflexive symmetric relation such that for every $(\nabla \vdash l = r) \in \mathsf{T}$, freshness context Δ, position C, permutation π, substitution θ, and fresh Γ (so if $a\#X \in \Gamma$ then $a \notin atms(\Delta, s, t)$),

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

We illustrate (1) and (2) with examples.

Example 3.5.
- Consider the theories $\vdash [a]X \to X$ and $\vdash [a]X = X$. We can show that $[b][a]a$ rewrites to $[a]b$ in the empty freshness context, that is, $\vdash_{\vdash [a]X \to X} [b][a]a \to [a]b$. For this, we first use reflexivity to transform $[b][a]a$ into $[a][b]b$ and then apply the rewrite rule at position $C = ([a]X, X)$. We can also show $\vdash_{\vdash [a]X = X} [b][a]a = [a]b$.
- Consider the rewrite theory $\beta\eta$ defining β- and η-reduction in the λ-calculus (see the Introduction). We can show that $\vdash_{\beta\eta} \mathsf{app}(\mathsf{lam}([a]\mathsf{app}(a,a)), b) \to \mathsf{app}(b,b)$ using rules $(\beta_{\mathtt{app}})$ and $(\beta_{\mathtt{var}})$.

3.1 Equivalence with the literature

The notions of equality and rewriting in (1) and (2) correspond to those in [GM09a] and [FG07] respectively. However, the presentation of (1) and (2) is original to this paper. Arguably, Definition 3.4 contains the clearest presentation of nominal rewriting and nominal algebra so far. It is certainly the most concise and it makes it easier to compare and constrast the two notions — to bring out what they have in common, and what is different.

Some checking needs to be done to verify that (1) and (2) coincide with nominal rewriting and nominal algebra as presented in the literature. All the main issues are indicated in the following two short sketches:

Remark 3.6 (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 π. This is because in [FG07] rewrite theories (Definition 3.1 in this paper) have the additional property that they be *equivariant* (Definition 4.2 of [FG07]). This means that if $R \in \mathsf{R}$ then $R^\pi \in \mathsf{R}$ (R^π is R with π 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 π in (1), and indeed, if $\Delta \vdash_{\mathsf{R}} s \to t$ then $\Delta \vdash_{\mathsf{R}} \pi \cdot s \to \pi \cdot t$.

Remark 3.7 (Nominal algebra)**.** (2) corresponds to Definition 3.10 and to the rules in Figures 1 and 2 in [GM09a]. The C corresponds to the congruence rules (**cong**[]) and (**cong**f); the π corresponds to the π in (**ax**) (modulo the same issue with r^π versus $\pi \cdot r$ mentioned in Remark 3.6); (**perm**) is built into $\approx_\alpha$.

Figure 2 of [GM09a] has an extra rule (**fr**), which generates a fresh atom. This corresponds to the fresh context Γ in (2). However, 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.

We inspect the rules in Figure 2 of [GM09a] and see that we can commute an instance of (**fr**) down through the other derivation rules; (**fr**) is a structural rule, which adds freshness assumptions and does not affect the subgoal to be proved.

If extra atoms in the derivation 'accidentally clash' with the atom generated by the instance of (**fr**), then we rename the fresh atom in the subderivation to be 'even fresher'. The reader familiar with the proof of weakening for first-order logic can recall how we rename the bound variable in the $\forall$-right rule to be fresh for the weakened context; the proof obligation here is identical and does not involve any 'nominal' elaborations.

Formally, an explicit inductive argument or the principle of ZFA equivariance [GP01] prove that fresh atoms do not matter up to renaming, so the renamed subderivation is still a valid subderivation. The interested reader is also referred to Lemma 5.10 in [GM08a] where a similar result is stated and proved in full detail, of a more complex system.

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

Theorem 4.2 and Theorem 4.4 describe how nominal rewriting relates to nominal algebra.

Definition 4.1. Suppose T is an equational theory and R is a rewrite theory. We say that R is a **presentation** of T if

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

We write $\Delta \vdash_{\mathsf{R}} s \leftrightarrow t$ for the symmetric closure $\Delta \vdash_{\mathsf{R}} s \to t$.

Proposition 4.2 (Soundness)**.** *Suppose* R *is a presentation of* T.
Then $\Delta \vdash_{\mathsf{R}} s \leftrightarrow t$ *implies* $\Delta \vdash_{\mathsf{T}} s = t$.

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

Suppose for some C, θ, and π,

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

Let $\Gamma = \varnothing$. 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 $(\mathbf{Axi}_{\nabla \vdash \mathbf{l}=\mathbf{r}})$. □

Remark 4.3. Suppose R is a presentation of T. It is not necessarily the case that $\Delta \vdash_{\mathsf{T}} s = t$ implies $\Delta \vdash_{\mathsf{R}} s \leftrightarrow t$. To see this, take $\mathsf{T} = \{a\#X \vdash X = \mathsf{f}(X)\}$ and $\mathsf{R} = \{a\#X \vdash X \to \mathsf{f}(X)\}$. Then $\vdash_{\mathsf{T}} X = \mathsf{f}(X)$ (using (**Axi**) with $\Gamma = a\#X$), but $\not\vdash_{\mathsf{R}} X \leftrightarrow \mathsf{f}(X)$.

Theorem 4.4 (Quasi-Completeness)**.** *Suppose* R *is a presentation of* T.
Then $\Delta \vdash_{\mathsf{T}} s = t$ *implies that there exists some fresh* Γ *(so if* $a\#X \in \Gamma$ *then* $a \notin atms(\Delta, s, t)$*) such that* $\Delta, \Gamma \vdash_{\mathsf{R}} s \leftrightarrow t$.

Note the 'fresh Γ' on the side of nominal rewriting.

Proof. We work by induction on the derivation of $\Delta \vdash_{\mathsf{T}} s = t$, write it Π.

The interesting case is $(\mathbf{Axi}_{\nabla \vdash \mathbf{l}=\mathbf{r}})$ for some $\nabla \vdash l = r \in \mathsf{T}$, of course. There is only one argument in the proof that is not obvious: Π is finite, so let us consider *all* the finitely many instances

of (**Axi**) in Π; write them $I_1, \ldots, I_n$. For each $1 \leq i \leq n$, I_i will involve $\nabla_i \vdash l_i = r_i$, C_i, π_i, θ_i, and a context Γ_i. (Note that Δ is constant across all these instances.)

Atoms in Γ_i do not feature in Δ, C_i, π_i, and θ_i — they are 'locally fresh'. However, they might 'accidentally' feature elsewhere in Π. It is a fact that because the atoms in Γ_i do not feature in Δ, C_i, π_i, and θ_i, they do not feature in the conclusion of I_i. Therefore, it is a fact that we can rename these atoms so that they are fresh for all parts of Π other than hypotheses of instances of (**Axi**), that is, there exists a derivation Π' of $\Delta \vdash_{\mathsf{T}} s = t$ such that for each $1 \leq i \leq n$ the respective Γ'_i in respective instances I'_i of (**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 Π'. This 'global freshness' condition is clearly preserved by taking subderivations. We now take $\Gamma = \bigcup_i \Gamma'_i$, and the proof is by a routine induction on Π'. Thus, an upper bound on Γ is the maximal size of the Γ_i.

Note that although ($\mathbf{Rew}_{\nabla \vdash \mathbf{l}=\mathbf{r}}$) appears to be more restrictive than ($\mathbf{Axi}_{\nabla \vdash \mathbf{l}=\mathbf{r}}$) in that (**Rew**) requires $s \equiv C[s']$ and s' α-equivalent to an instance of a left-hand side, this is not an issue because the rewrite relation is transitive and includes the α-equivalence relation. □

5 Closed rewriting and nominal algebra

Theorem 4.4 establishes a completeness result for nominal rewriting modulo additional freshness constraints (the extra Γ).

This mismatch between nominal rewriting and nominal algebra could be solved by including fresh atom generation in the definition of a rewriting step. But this comes at a cost — the freshness context may change along a rewrite derivation, and with it also the notion of α-equivalence — and is not needed for large classes of systems, as we show below.

In this section, we show that closed nominal rewriting is complete for nominal algebra equality when all the axioms are *closed*.

Although there are interesting systems, such as the axiomatisation of the π-calculus [FG05, FG07], which are not closed, this result has many applications: all the systems that arise from functional programming (including the axiomatisation of the λ-calculus) are closed, and all the systems that can be specified in a standard higher-order rewriting formalism are also closed (see [FG05]).

5.1 The definition of closed rules and closed rewriting

Definition 5.1 (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

$$(\nabla \vdash l) \stackrel{?}{\approx} (\Delta \vdash s) \quad \text{where } \mathit{unkn}(\nabla \vdash l) \cap \mathit{unkn}(\Delta \vdash s) = \varnothing.$$

A **solution** to this problem is a substitution σ such that

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

Remark 5.2. Nominal matching is decidable [UPG04], and can be solved in linear time [CF08].

Definition 5.3 (Freshened variants)**.** If t is a term, we say that $t^\prime$ is a **freshened variant** of t when $t^\prime$ has the same structure as t, except that the atoms and unknowns have been replaced by 'fresh'

atoms and unknowns (so they are not in $atms(t)$ and $unkn(t)$, and perhaps are also fresh with respect to some atoms and unknowns from other syntax, which we will always specify). We omit an inductive definition.

Similarly, if ∇ is a freshness context then ∇^ν will denote a freshened variant of ∇ (so if $a\#X \in \nabla$ then $a^\nu\#X^\nu \in \nabla^\nu$, where a^ν and X^ν are chosen fresh for the atoms and unknowns appearing in ∇).

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

Note that if $\nabla^\nu \vdash l^\nu \to r^\nu$ is a freshened variant of $\nabla \vdash l \to r$ then $unkn(\nabla^\nu \vdash l^\nu \to r^\nu) \cap unkn(\nabla \vdash l \to r) = \varnothing$.

Example 5.4. For example:

- $[a^\nu][b^\nu]X^\nu$ is a freshened variant of $[a][b]X$, $a^\nu\#X^\nu$ is a freshened variant of $a\#X$, and $\varnothing \vdash a^\nu \to b^\nu$ is a freshened variant of $\varnothing \vdash a \to b$.
- Neither $[a^\nu][a^\nu]X^\nu$ nor $[a^\nu][b^\nu]X$ 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 5.5. A term-in-context $\nabla \vdash l$ is **closed** if there exists a solution for the matching problem

$$(\nabla^\nu \vdash l^\nu) \quad {}_?{\approx} \quad (\nabla, atms(\nabla^\nu, l^\nu)\#unkn(\nabla, l) \vdash l). \tag{3}$$

Lemma 5.6. *$\nabla \vdash l$ is closed when there exists a substitution σ with $dom(\sigma) \subseteq unkn(\nabla^\nu \vdash l^\nu)$ such that $\nabla, atms(\nabla^\nu, l^\nu)\#unkn(\nabla, l) \vdash (\nabla^\nu\sigma,\ l \approx_\alpha l^\nu\sigma)$.*

Definition 5.7.

- Call $R = (\nabla \vdash l \to r)$ and $A = (\nabla \vdash l = r)$ **closed** when $\nabla \vdash (l, r)$ is closed[2].
- Given a rewrite rule $R = (\nabla \vdash l \to r)$ and a term-in-context $\Delta \vdash s$, write $\Delta \vdash s \overset{R}{\to}_c t$ when there is some R^ν a freshened variant of R (so fresh for R, Δ, s, and t), position C and substitution θ such that

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

  We call this (one-step) **closed rewriting**.

  The *closed rewrite relation* $\Delta \vdash_R s \to_c t$ is the reflexive transitive closure as in Definition 3.4.

The choice of freshened variant of $\nabla \vdash l$ in Definition 5.5 does not matter. Similarly for closed rewriting in Definition 5.7. This is related to the some/any property of the $\mathcal{N}$-quantifier [GP01], and to the principle of ZFA equivariance described e.g. in [GM09a, Theorem A.4]. One way to look at Definitions 5.5 and 5.7 is that the atoms in $\nabla^\nu \vdash l^\nu$ occupy a 'separate namespace'.

Remark 5.8. Closed nominal terms and rewriting were introduced in [FGM04]. $\Delta \vdash s \overset{R}{\to}_c t$ when s rewrites to t using a version of R where the atoms and unknowns are renamed to be fresh. Renaming unknowns to be fresh is standard in rewriting, where variables in a rewrite rule are assumed distinct from those of the terms to be rewritten. What is special about closed rewriting is that it applies a similar renaming to the atoms.

So for example, $\vdash a \overset{a\to b}{\to} b$ and $\vdash c \overset{a\to b}{\to} d$, but $\not\vdash a \overset{a\to b}{\to}_c b$ and $\not\vdash c \overset{a\to b}{\to} d$.

A rule R is closed when, intuitively, it is equal to any freshened variant R^ν up to a substitution. $a \to b$ is not closed; the rules in [FG07] for λ-calculus β-reduction are closed; those for π-calculus reduction are not closed.

[2]Here we use pair as a term former and apply the definition above.

Comparing Definition 5.7 (closed rewriting) with Definition 3.4 (rewriting) we see they are very similar. However, there are two key differences:

- The π in (1) in Definition 3.4 is not there in (4) in Definition 5.7. This π can be very expensive [Che04], so removing it greatly increases the efficiency of calculating closed nominal rewrites.
- Atoms cannot 'interact by name' in a closed rewrite step, because they are renamed.

5.2 Properties of closed rewriting, and connection with nominal algebra

First we will prove a strengthening property for closed rewriting, for which we need some preliminary lemmas.

Definition 5.9. We define the substitution $\sigma \circ \pi$ by:

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

Lemma 5.10. *If $atms(s) \cap nontriv(\pi) = \varnothing$ then $\pi\cdot(s\sigma) \equiv s(\sigma\circ\pi)$.*

Lemma 5.11.
1. *Suppose $a \notin atms(s', l^{\nu})$. Then if $\Delta \vdash s' \approx_\alpha l^{\nu}\sigma$ then there exists σ' such that $\Delta \vdash \sigma(X) \approx_\alpha \sigma'(X)$ and $a \notin atms(\sigma'(X))$, for all $X \in unkn(l^{\nu})$.*
2. *Suppose $a \notin atms(t, r^{\nu}, C)$. Then if $\Delta \vdash C[r^{\nu}\sigma] \approx_\alpha t$ then there exists some σ' such that $\Delta \vdash \sigma(X) \approx_\alpha \sigma'(X)$ and $a \notin atms(\sigma'(X))$, for all $X \in unkn(r^{\nu})$.*

Proof. For the first part, we construct σ' by an induction on the structure of l^{ν}. We sketch one case:

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

For the second part we work by induction on the derivation of $\Delta \vdash C[r^{\nu}\sigma] \approx_\alpha t$, using the rules in Figure 1 to break down C until we reach the first case (note that $\approx_\alpha$ is symmetric). □

Lemma 5.12. *If $\Delta \vdash \sigma(X) \approx_\alpha \sigma'(X)$ for all $X \in unkn(t)$ then $\Delta \vdash t\sigma \approx_\alpha t\sigma'$.*

Proposition 5.13 (Strengthening for closed rewriting)**.** *Fix a context Δ and terms s and t. Suppose Γ is fresh (so if $a\#X \in \Gamma$ then $a \notin atms(s,t,\Delta)$). Suppose $R = (\nabla \vdash l \to r)$ 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, there is some freshened R^{ν} (with respect to s, t, Δ,Γ, and R), and some position C and substitution σ such that $dom(\sigma) \subseteq unkn(R^{\nu})$ and

$$s \equiv C[s'] \qquad \Delta,\Gamma, atms(R^{\nu}) \# unkn(\Delta,s,t) \vdash (\nabla^{\nu}\sigma,\ s' \approx_\alpha l^{\nu}\sigma,\ C[r^{\nu}\sigma] \approx_\alpha t).$$

Using Lemmas 5.11 and 5.12 we may assume without loss of generality that $a \notin atms(\sigma)$. By elementary calculations on the atoms of terms and using Strengthening (Lemma 2.11) we deduce

$$s \equiv C[s'] \qquad \Delta, atms(R^{\nu}) \# unkn(\Delta,s,t) \vdash (\nabla^{\nu}\sigma,\quad s' \approx_\alpha l^{\nu}\sigma,\quad C[r^{\nu}\sigma] \approx_\alpha t).$$

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 2.13). □

We now establish the relationship between nominal rewriting and closed rewriting. The first result, Proposition 5.15 below, shows that when a rule is closed, nominal rewriting implies closed rewriting (this result was first shown as part of [FG07, Theorem 70]; we give a shorter proof here). The second result, Proposition 5.17 below relating a closed rewriting step with a nominal rewrite step, is new and is the key to obtain a completeness proof for closed rewriting with respect to nominal algebra.

Lemma 5.14. *$\Delta \vdash a\#s$ if and only if $\Delta \vdash \pi(a)\#\pi \cdot s$.*

Proposition 5.15. *If $R = (\nabla \vdash l \rightarrow r)$ is closed then $\Delta \vdash s \xrightarrow{R} t$ implies $\Delta \vdash s \xrightarrow{R}_c t$.*

Proof. Suppose $\Delta \vdash s \xrightarrow{R} t$. So there exist Δ, C, s', π, and θ such that

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

Without loss of generality we assume $unkn(\theta(X)) \subseteq unkn(\Delta, s, t)$ for every $X \in dom(\theta)$ (because we only 'use' the part of θ that maps l to s and r to t).

$\nabla \vdash l \rightarrow r$ is closed so by Lemma 5.6 there is a freshened variant $R^\nu = (\nabla^\nu \vdash l^\nu \rightarrow r^\nu)$ of R and a substitution σ such that $dom(\sigma) \subseteq unkn(R^\nu)$ and

$$\nabla, atms(l^\nu) \# unkn(\Delta, s, t) \vdash (\nabla^\nu\sigma, \quad l \approx_\alpha l^\nu\sigma, \quad r \approx_\alpha r^\nu\sigma).$$

It is not hard to use our assumptions to verify that

$$\Delta, atms(l^\nu) \# unkn(\Delta, s, t) \vdash atms(l^\nu) \# unkn(\Delta, s, t)\theta.$$

It follows using Lemmas 2.13 and 2.6 that

$$s \equiv C[s'] \qquad \Delta, atms(l^\nu) \# unkn(\Delta, s, t) \vdash \big(\nabla^\nu\sigma\theta,\ s' \approx_\alpha \pi \cdot (l^\nu\sigma\theta),\ C[\pi \cdot (r^\nu\sigma\theta)] \approx_\alpha t\big).$$

By assumption the atoms in R^ν are fresh and so we can assume $atms(R^\nu) \cap nontriv(\pi) = \varnothing$. It follows by Lemmas 5.10 and 2.7 that $\pi \cdot (l^\nu\sigma\theta) \equiv l^\nu((\sigma \circ \theta) \circ \pi)$ and $\pi \cdot (r^\nu\sigma\theta) \equiv r^\nu((\sigma \circ \theta) \circ \pi)$. Using Lemma 5.14 $\Delta, atms(l^\nu)\#unkn(\nabla, l) \vdash \nabla^\nu((\sigma \circ \theta) \circ \pi)$ also follows. Write θ' for $(\sigma \circ \theta) \circ \pi$. Then

$$s \equiv C[s'] \qquad \Delta, atms(l^\nu) \# unkn(\Delta, s, t) \vdash \big(\nabla^\nu\theta', \quad s' \approx_\alpha l^\nu\theta', \quad C[r^\nu\theta'] \approx_\alpha t\big).$$

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

Lemma 5.16. *Suppose $\nabla \vdash l$ is a closed term-in-context where $atms(\nabla \vdash l) = \{a_1, \ldots, a_n\}$ and $unkn(\nabla \vdash l) = \{X_1, \ldots, X_n\}$; we take these atoms and unknowns in some fixed but arbitrary order.*

Suppose is $\nabla^\nu \vdash l^\nu$ a freshened variant of $\nabla \vdash l$ where $atms(\nabla^\nu \vdash l^\nu) = \{a_1^\nu, \ldots, a_n^\nu\}$ and $unkn(\nabla^\nu \vdash l^\nu) = \{X_1^\nu, \ldots, X_n^\nu\}$; we take these fresh atoms and unknowns in a corresponding order.

Let τ and ς be the permutation and substitution defined by

$$\tau = (a_1^\nu\ a_1) \circ \ldots \circ (a_n^\nu\ a_n) \quad \text{and} \quad \varsigma = [X_1 \mapsto \tau \cdot X_1^\nu, \ldots, X_n \mapsto \tau \cdot X_n^\nu],$$

then:

1. $l^\nu \equiv \tau \cdot (l\varsigma)$.

2. $\Gamma' \vdash \nabla^{\nu}\theta$ *if and only if* $\Gamma' \vdash \nabla\varsigma\theta$.

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

$$\tau \cdot ((\pi \cdot X)\varsigma) \overset{\text{Lemma 2.5}}{\equiv} (\tau \circ \pi) \cdot \varsigma(X) \overset{\text{Lemma 2.5}}{\equiv} (\tau \circ \pi \circ \varsigma) \cdot X \overset{\text{fact}}{\equiv} \pi' \cdot X^{\nu}.$$

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

$$X^{\nu}\theta \overset{\text{Lemma 2.5}}{\equiv} (\tau \cdot (X\varsigma))\theta \overset{\text{Lemma 2.6}}{\equiv} \tau \cdot (X\varsigma\theta).$$

By Lemma 5.14 $\Gamma' \vdash a^{\nu}\#(X^{\nu}\theta)$ if and only if $\Gamma' \vdash a\#(X\varsigma\theta)$. The result follows. □

Proposition 5.17. *If* $R = (\nabla \vdash l \to r)$ *is closed then* $\Delta \vdash s \overset{R}{\to}_c t$ *implies there is some fresh* Γ *(so if* $a\#X \in \Gamma$ *then* $a \notin atms(\Delta, s, t)$*) such that* $\Delta, \Gamma \vdash s \overset{R}{\to} t$.

Proof. If $\Delta \vdash_{\mathsf{R}} s \to_c t$ then for some freshened variant $R^{\nu} = (\nabla^{\nu} \vdash l^{\nu} \to r^{\nu})$ of R (freshened with respect to R, Δ, s, and t) there exists some position C, term s', and substitution θ such that

$$s \equiv C[s'] \qquad \Delta, atms(R^{\nu}) \# unkn(\Delta, s, t) \vdash (\nabla^{\nu}\theta, \quad s' \approx_{\alpha} l^{\nu}\theta, \quad C[r^{\nu}\theta] \approx_{\alpha} t).$$

By Lemmas 5.16 and 2.6, there exists τ and ς such that:

$$s \equiv C[s'] \qquad \Delta, atms(R^{\nu}) \# unkn(\Delta, s, t) \vdash (\nabla\varsigma\theta,\ s' \approx_{\alpha} \tau \cdot (l\varsigma\theta),\ C[\tau \cdot (r\varsigma\theta)] \approx_{\alpha} t).$$

Using Lemmas 2.7 and 2.6 we deduce $\Delta, atms(R^{\nu}) \# unkn(\Delta, s, t) \vdash s \overset{R}{\to} t$. □

Definition 5.18. $\Delta \vdash_{\mathsf{R}} s \leftrightarrow_c t$ denotes the symmetric closure of $\Delta \vdash_{\mathsf{R}} s \to_c t$.

Theorem 5.19 (Soundness and completeness)**.** *Suppose the rewrite theory* R *is a presentation (Definition 4.1) of the equational theory* T*. Suppose all rules in* R *are closed. Then* $\Delta \vdash_{\mathsf{T}} s = t$ *if and only if* $\Delta \vdash_{\mathsf{R}} s \leftrightarrow_c t$.

Proof. Suppose $\Delta \vdash_{\mathsf{T}} s = t$. By Theorem 4.4 there is a fresh Γ such that $\Delta, \Gamma \vdash_{\mathsf{R}} s \leftrightarrow t$. By Proposition 5.15 and Strengthening (Proposition 5.13) $\Delta \vdash_{\mathsf{R}} s \leftrightarrow_c t$.

Conversely, suppose $\Delta \vdash_{\mathsf{R}} s \leftrightarrow_c t$. By Proposition 5.17 $\Delta, \Gamma \vdash_{\mathsf{R}} s \leftrightarrow t$ for some fresh Γ. It follows by Proposition 4.2 that $\Delta \vdash_{\mathsf{T}} s = t$. □

5.3 Mechanising equational reasoning

Closed nominal rewriting can be used to automate reasoning in nominal equational theories, provided that the theory satisfies certain conditions.

Definition 5.20. A rewrite theory R is **closed** when every $R \in \mathsf{R}$ is closed (Definition 5.7). We say that t is an **(R-)normal form** of s if $\Delta \vdash_{\mathsf{R}} s \to_c t$ and there is no u such that $\Delta \vdash_{\mathsf{R}} t \to_c u$ (so there is no rewrite from t).

A theory R is **terminating** when there are no infinite closed rewriting sequences $\Delta \vdash_{\mathsf{R}} t_1 \to_c t_2,\ t_2 \to_c t_3,\ \ldots$. It is **confluent** when, if $\Delta \vdash_{\mathsf{R}} s \to_c t$ and $\Delta \vdash_{\mathsf{R}} s \to_c t'$, then u exists such that $\Delta \vdash_{\mathsf{R}} t \to_c u$ and $\Delta \vdash_{\mathsf{R}} t' \to_c u$.

A theory R is **convergent** when it is terminating and confluent.

Theorem 5.21. *Suppose the axioms in a theory* T *can be oriented to form a closed* R*. If* R *is confluent, then* $\Delta \vdash_{\mathsf{T}} s = t$ *if and only if there exists* u *such that* $\Delta \vdash_{\mathsf{R}} s \to_c u$ *and* $\Delta \vdash_{\mathsf{R}} t \to_c u$*.*

Proof. By Theorem 5.19. □

Theorem 5.21 does not require termination. If we have termination then we can decide whether there exists a term u with the desired property: it suffices to rewrite s and t to normal form and then check that the normal forms are α-equivalent (convergence guarantees existence and unicity of normal forms up to α-equivalence; a linear-time algorithm to check α-equivalence is described in [CF09]). Also, since Theorem 5.21 uses closed rewriting, the computation of a rewrite step is efficient: nominal matching is sufficient (see also [CF09] for linear-time nominal matching algorithms).

Corollary 5.22 (Decidability of deduction in T)**.** *Suppose* T *is an equational theory whose axioms can be oriented to form a closed* R*. Suppose* R *is convergent. Then equality is decidable in* T *(i.e.,* $\Delta \vdash_{\mathsf{T}} s = t$ *is a decidable relation).*

6 Conclusions

Efficient algorithms for closed nominal rewriting and for checking α-equivalence are described in [CF09]. We can also check that rules are closed in linear time, with the nominal matching algorithm of [CF09]. It follows from Corollary 5.22 that, had we a procedure to check that a given set of rules is convergent, we could directly build 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 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 α-equivalence, which might simplify the proofs here. Permissive nominal terms have been implemented in prototype form [Mul09], but it remains to consider more efficient algorithms 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, 1998.

[Cal10] Christophe Calvès. Complexity and implementation of nominal algorithms, 2010. PhD thesis, King's College London.

[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.

[CP07] Ranald A. Clouston and Andrew M. Pitts. Nominal equational logic. In L. Cardelli, M. Fiore, and G. Winskel, editors, *Computation, Meaning and Logic. Articles dedicated to Gordon Plotkin*, volume 1496 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2007.

[CU08] James Cheney and Christian Urban. Nominal logic programming. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(5):1–47, 2008.

[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.

[DGM10] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification (journal version). *Logic Journal of the IGPL*, 2010. In press.

[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.

[DPS97] Joëlle Despeyroux, Frank Pfenning, and Carsten Schürmann. Primitive recursion for higher-order abstract syntax. In Philippe de Groote and J. Roger Hindley, editors, *Proc. Int'l Conf. on Typed Lambda Calculi and Applications (TLCA'97)*, volume 1210 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 1997. An extended version is available as CMU Technical Report CMU-CS-96-172.

[FG05] Maribel Fernández and Murdoch J. Gabbay. Nominal Rewriting with Name Generation: Abstraction vs. Locality. In *Proceedings of the 7th ACM-SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP'05), Lisbon, Portugal*. ACM Press, 2005.

[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)*. 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.

[GM06] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. In *18th Nordic Workshop on Programming Theory*, 2006.

[GM08a] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM08b] 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. *Journal of Logic and Computation*, 19(6):1455–1508, December 2009.

[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+92] Joseph Goguen, Andrew Stevens, Hendrik Hilberdink, Keith Hobley, W. A. Hunt, and T. F. Melham. 2obj: A metalogical framework theorem prover based on equational logic. *Philosophical Transactions: Physical Sciences and Engineering*, 1992.

[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.

[LV08] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. In *Rewriting Techniques and Applications, Proceedings of RTA 2008*, number 5117 in Lecture Notes in Computer Science. Springer, 2008.

[LV10] Jordi Levy and Mateu Villaret. An efficient nominal unification algorithm, 2010. In *Rewriting Techniques and Applications, Proceedings of RTA 2010*.

[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.

[Mil91] Dale Miller. Unification of simply typed lambda-terms as logic programming. In *Eighth International Logic Programming Conference*, pages 255–269. MIT Press, 1991.

[MN98] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, II. *Information and Computation*, 100(1):41–77, September 1992.

[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.

[PE88] Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *PLDI (Programming Language Design and Implementation)*, pages 199–208. ACM Press, 1988.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal Unification. *Theoretical Computer Science*, 323(1–3):473–497, 2004.