Unity in nominal equational reasoning: the algebra of equality on nominal sets

Murdoch J. Gabbay

Abstract

There are currently no fewer than four dedicated logics for equality reasoning over nominal sets: nominal algebra, nominal equational logic, nominal equational logic with equality only, and permissive-nominal algebra.

In this survey and research paper we present these logics side-by-side in a common notation, survey their similarities and differences, discuss their proof- and model-theories, and discuss in detail what the implications of those differences are for mathematical reasoning in each of them.

Keywords: Nominal algebra, nominal equational reasoning, equality logic, nominal sets, (permissive-)nominal terms, semantic and syntactic freshness.

Contents

1. Introduction

1.1. Nominal sets for names and binding

Nominal sets, introduced in [GP01], are a particularly abstract and effective way of giving semantics to names and binding (for a definition see Section 4). Suppose you want to axiomatise capture-avoiding substitution $[a{:=}t]$ over a calculus (a logic or rewrite system) with a binder—call it, say, λ. Then you will almost certainly write this:

$$(\lambda x.r)[y{:=}t] = \lambda x.(r[y{:=}t]) \quad \text{if } x \text{ is not free in } t$$

You might also write 'choose y fresh and α-rename $\lambda x.r$ to $\lambda y.(r[y/x])$'.

Nominal sets were developed to interpret these kinds of assertions [GP01]; names, binding, fresh choice of names, and α-renaming become primitives of the denotation. They are represented directly, rather than e.g. as *de Bruijn indexes* [dB72], *functional abstraction* [PE88], or Bourbaki's *boxes and links* notation [Bou70, Section 1].

Once nominal sets are established it is only a matter of time before we develop term and predicate languages to describe them. The term language came in the form of nominal terms [UPG04]. The first predicate language for nominal terms was actually *nominal rewriting* [FG07] but the first 'logical' language (focussed on derivability) was nominal algebra—followed by nominal equational logic, then permissive-nominal algebra and nominal equational logic with equality only (also permissive-nominal logic, which adds quantifiers [DG10, DG12]).

In nominal algebra and nominal equational logic, the two fragments above can be represented as follows (nominal rewriting would be just the same, but with $\rightarrow$ instead of $=$):

- $a\#X \vdash (\lambda[a]Z)[b{\mapsto}X] = \lambda[a](Z[b{\mapsto}X])$
- $b\#X \vdash \lambda[b](b\ a){\cdot}X = \lambda[a]X$.

The permissive-nominal version replaces the freshness condition '$a\#X$' with a typing condition that $a \notin \mathbb{A}^<$, where $\mathbb{A}^<$ and $\mathbb{A}^>$ partition $\mathbb{A}$ into two infinite subsets:

- $(\lambda[a]Z)[b{\mapsto}X] = \lambda[a](Z[b{\mapsto}X])$ (where $a \notin \mathbb{A}^<$ and $b \in \mathbb{A}^<$).
- $\lambda[b](b\ a){\cdot}X = \lambda[a]X$ (where $b \notin \mathbb{A}^<$).

See Subsection 3.4 in this paper or [DG10, DG12, DGM09, DGM10].[1]

Details follow; our point here is that this family of logics is designed to represent, more-or-less symbol-for-symbol, informal reasoning on specifications with binders. Here are some examples, just to prove that such specifications occur and are useful: quantifiers in logic ($\forall$), functional abstraction in the λ-calculus (λ), integration ($\int$), and name-restriction in the π-calculus (ν).

Names and binding matter, and logics for equality over nominal sets matter because equality reasoning is prototypical for serious and usable meta-reasoning systems.

1.2. Four logics for equality

A glance at the literature reveals no fewer than four logics for equational reasoning over nominal sets:

- *nominal algebra* (NA) as presented e.g. in [GM06a, GM06b, GM06c, GM07, GM09a],
- *nominal equational logic* (NEL) as presented e.g. in [CP07, Clo09],
- NEL *with equality only* [Clo09, Clo11], and

[1] The advantage of the permissive syntax is that a mutable freshness context is replaced by a static typing condition and permissive-nominal syntax can be directly quotiented by α-equivalence.

- *permissive-nominal algebra* (PNA) [GM09b, Gab12b].[2]

This raises the question of why we need these logics, what their distinctive features are, how they can be applied, and to what extent they are saying the same thing.

In fact, these logics are all just different ways of approaching the same mathematical entity, much as there are many different and equivalent choices of connectives for setting up first-order logic. They do differ in complexity; which in decreasing order are (according to this author, at least): NEL, NEL with equality only, NA, and PNA.

It is quite hard to get an overview of this design space from individual research papers, since these tend to concentrate on the demands of the particular theorems at hand. Thus there seems to be a space in the literature for a paper dedicated specifically to presenting what equational reasoning over nominal sets looks like, to chart the development of the various logics, and to discuss how the author's own work on NA and PNA fits in to the larger picture.

The discussion and theorems in this paper do not just matter for this family of logics. The design choices being made here may be prototypical for the design of any logic over nominal sets; these choices should be informed by a rich and detailed understanding of the design space, to which we hope this paper can contribute.

1.3. Structure of the paper

The structure of the paper is as follows:

- In Section 2 we introduce nominal terms.
- In Section 3 we present nominal algebra, nominal equational logic, nominal equational logic with equality only, and permissive-nominal algebra, and in Subsection 3.5 we give a first discussion of them.
- Denotations in nominal sets are considered in Section 4, and these are used as a mathematical foundation for a discussion of semantic freshness in nominal algebra and translation of NEL to NA in Section 5.
- Section 6 includes an extended and detailed discussion of the design issues involved in designing algebra over nominal sets, with a particular focus on how Clouston and Pitts's evaluation of NA has been consistently wide of the mark, from [CP07] to [Clo11].
- Section 7 focuses on the more recent *permissive-nominal algebra* by this author with Dowek and Mulligan. This is a more recent system and we do not give many proofs, focussing instead on giving some useful idea of how the permissive-nominal techniques slot in to the broader territory and why it was developed.
- Section 8 more briefly discusses a key feature of nominal techniques, namely, the ability to talk about name-abstraction.
- We conclude with brief sketches of two other 'nominal' logics in Section 9 and a concluding discussion in Section 10.

2. Nominal terms

Definition 2.1. For each natural number $i \in \mathbb{N} = \{0, 1, 2, \dots\}$ fix a disjoint **set of atoms** $\mathbb{A}_i$. Let a, b, c range over distinct atoms.[3]

Definition 2.2. Fix a **signature** $\Sigma = (AtomicSort, BaseSort, TermFormer, ar)$ which is a tuple of a set of **atomic sorts** $\nu \subseteq \mathbb{N}$, **base sorts** τ, a set of **term-formers** f, and an **arity** function mapping each f to a tuple $ar(\mathsf{f}) = (\alpha_1, \dots, \alpha_n)\tau$ where α_i range over atomic or base sorts.

[2] Cousins of these logics are the *Synthetic Nominal Equational Logic* of Fiore and Hur [FH08] and the presentation of nominal terms as many-sorted first-order terms by Kurz and Petrişan [KP10], which we shall also consider, briefly, in Section 9.

[3] So a, b, c are meta-variables ranging over distinct atoms. Call this the *permutative convention* (which goes back to [GM06c, GM08]); it accurately models what we mean when we write e.g. $\lambda x.\lambda y.xy$.

f, g will range over distinct term-formers. α will range over atomic or base sorts, or just **sorts** for short.

For each sort α fix a countably infinite set of **variables** of that sort. X, Y, Z will range over distinct variables and we write $sort(X)$ for the sort of X.

Remark 2.3. Elsewhere, such as in [UPG04, FG07, GM09a, CP07, Clo11], *atoms-abstraction* sorts $[\nu]\tau$ are also considered. As noted in [GM09a] and [Clo10] this adds no expressivity, so we omit abstraction sorts here. This is only for simplicity.

Definition 2.4. A **permutation** is a bijection on atoms such that $nontriv(\pi) = \{a \mid \pi(a) \neq a\}$ is finite and $\pi(a) \in \mathbb{A}_\nu$ if and only if $a \in \mathbb{A}_\nu$. Write $\mathbb{P}$ for the set of all permutations.

Write id for the identity permutation and $\circ$ for composition of permutations. Write π^{-1} for inverse.

Definition 2.5. Nominal terms are inductively defined by:

$$r ::= a \mid \pi{\cdot}X \mid \mathsf{f}(r, \ldots, r)$$

We define a **sorting relation** $r : \alpha$ by:

$$\frac{(a \in \mathbb{A}_\nu)}{a : \nu} \qquad \frac{(sort(X) = \alpha)}{\pi{\cdot}X : \alpha} \qquad \frac{r_1 : \alpha_1 \cdots r_n : \alpha_n \quad (ar(\mathsf{f}) = (\alpha_1, \ldots, \alpha_n)\tau)}{\mathsf{f}(r_1, \ldots, r_n) : \tau}$$

Notation 2.6. It is not hard to show that sorts are unique where they exist. Write $sort(r) = \alpha$ when $r : \alpha$ and call r **well-sorted** when $r : \alpha$ for some α. We are only interested in well-sorted terms, so henceforth we restrict to well-sorted terms.

Remark 2.7. a is an atom representing a name (e.g. an object-level variable symbol, a channel name, a memory location, and so on). $\pi{\cdot}X$ is a pair of π and X and is called a **moderated unknown**; and $\mathsf{f}(r_1, \ldots, r_n)$ is a term-former applied to terms.

Example 2.8. Assume one atomic type ν, one base type τ, and term-formers var, app, and lam where $ar(\mathsf{var}) = (\nu)\tau$, $ar(\mathsf{app}) = (\tau, \tau)\tau$, and $ar(\mathsf{lam}) = (\nu, \tau)\tau$.

Terms of the untyped λ-calculus can be represented by $\mathsf{var}(a)$ (variable symbols), $\mathsf{app}(r', r)$ (application), and $\mathsf{lam}(a, r)$ (lambda-abstraction), all of which are terms of sort τ. See also Subsection 3.5 (a theory of functions).

Definition 2.9. Define a **permutation action** on a nominal term as follows:

$$\pi{\cdot}a = \pi(a) \qquad \pi{\cdot}(\pi'{\cdot}X) = (\pi \circ \pi'){\cdot}X \qquad \pi{\cdot}\mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}(\pi{\cdot}r_1, \ldots, \pi{\cdot}r_n)$$

Definition 2.10. A **substitution** θ is a map from unknowns to terms. Give terms a **substitution action** as follows:

$$a\theta = a \qquad (\pi{\cdot}X)\theta = \pi{\cdot}\theta(X) \qquad \mathsf{f}(r_1, \ldots, r_n)\theta = \mathsf{f}(r_1\theta, \ldots, r_n\theta)$$

Definition 2.11. Define a notion of **variables** of a nominal term[4] as follows:

$$vars(a) = \varnothing \qquad vars(\pi{\cdot}X) = \{X\} \qquad vars(\mathsf{f}(r_1, \ldots, r_n)) = \textstyle\bigcup_1^n vars(r_i)$$

Definition 2.12. A **freshness context** Δ is a finite set of **freshness assumptions** $a\#X$.

3. Equality reasoning over nominal terms

3.1. Nominal algebra

We need some notation to help us express the notion of derivable equalities in nominal algebra (Definition 3.6).

[4] Our nominal terms syntax has no binder for X. For that, see permissive-nominal logic [DG10, Gab12b].

$$\frac{}{\mathsf{T};\Delta \vdash_{\text{NA}} r = r}\ (\textbf{Refl}) \qquad \frac{\mathsf{T};\Delta \vdash_{\text{NA}} r = s \quad \mathsf{T};\Delta \vdash_{\text{NA}} s = t}{\mathsf{T};\Delta \vdash_{\text{NA}} r = t}\ (\textbf{Trans})$$

$$\frac{\mathsf{T};\Delta \vdash_{\text{NA}} r = s}{\mathsf{T};\Delta \vdash_{\text{NA}} s = r}\ (\textbf{Symm}) \qquad \frac{(a \notin fa_{\Delta}(r),\ \ b \notin fa_{\Delta}(r))}{\mathsf{T};\Delta \vdash_{\text{NA}} (a\ b)\cdot r = r}\ (\textbf{Perm})$$

$$\frac{\big(\forall X.fa_{\Delta'}(\theta(X)) \subseteq fa_{\Delta}(X),\ (\Delta \vdash r{=}s){\in}\mathsf{T}\big)}{\mathsf{T};\Delta' \vdash_{\text{NA}} \pi\cdot(r\theta) = \pi\cdot(s\theta)}\ (\mathbf{Ax}_{\mathbf{\Delta\vdash r=s}}) \qquad \frac{\mathsf{T};\Delta \vdash_{\text{NA}} r_i = s_i \qquad (1 \le i \le n)}{\mathsf{T};\Delta \vdash_{\text{NA}} \mathsf{f}(r_1,\ldots,r_n) = \mathsf{f}(s_1,\ldots,s_n)}\ (\textbf{Cong})$$

$$\frac{\mathsf{T};\Delta, a\#X \vdash_{\text{NA}} r = s\ \ (a{\notin}atms(r,s))}{\mathsf{T};\Delta \vdash_{\text{NA}} r = s}\ (\textbf{Fresh})$$

Figure 1: Derivable entailment in Nominal Algebra (NA)

Definition 3.1. If $A \subseteq \mathbb{A}$ is a set of atoms, define $\pi\cdot A = \{\pi(a) \mid a \in A\}$.

Definition 3.2. Define a notion of **free atoms** of a nominal term (in a freshness context) $fa_{\Delta}(r)$ as follows:

$$fa_{\Delta}(a) = \{a\} \qquad fa_{\Delta}(\pi\cdot X) = \pi\cdot\{a \mid a\#X{\notin}\Delta\} \qquad fa_{\Delta}(\mathsf{f}(r_1,\ldots,r_n)) = \bigcup_1^n fa_{\Delta}(r_i)$$

'Free atoms of' is parameterised over Δ. For example,

$$fa_{\{a\#X,b\#X,c\#Y\}}(X) = \mathbb{A} \setminus \{a,b\}.$$

Remark 3.3. In [UPG04, FG07] and elsewhere, the judgement $a \notin fa_{\Delta}(r)$ is written $\Delta \vdash a\#r$. This is purely a matter of notation; see Subsection 6.1 for a further discussion of this.

We need one more freshness definition; in [GM09a] it is written $a \notin r$ and in [CP07] it is written $a\#r$:

Definition 3.4. Define $atms(r)$ the atoms **appearing in** r inductively by:

$$atms(a) = \{a\} \qquad atms(\pi\cdot X) = nontriv(\pi) \qquad atms(\mathsf{f}(r_1,\ldots,r_n)) = \bigcup_1^n atms(r_i)$$

Remark 3.5. Having two freshness judgements, $a \notin fa_{\Delta}(r)$ and $a \notin atms(r)$, is an artefact of using a 'primitive' name-carrying syntax. In permissive-nominal techniques as presented in [Gab12b], $a \notin fa_{\Delta}(r)$ and $a \notin atms(r)$ coincide. So by promoting permissive-nominal techniques, this author is trying to move the literature towards a world in which Definitions 3.2 and 3.4 are conflated (cf. Remark 6.4). Since this paper is partly a survey, we have to faithfully represent the tools we used at the time.

Definition 3.6. A **nominal equality** E is a tuple $\Delta \vdash l = r$.

A **nominal algebra theory** T is a signature along with a set of nominal equalities in that signature.

The **derivable equalities** $\mathsf{T};\Delta \vdash_{\text{NA}} r = s$ are generated by the rules in Figure 1.

3.2. Nominal equational logic

Nominal equational logic adds a *semantic freshness judgement*.

Definition 3.7. A **nominal freshness** is a tuple $\Delta \vdash a\#r$. A **nominal freshness axiom** is a nominal freshness. A **nominal equational logic** theory U is a signature along with a set of freshness or equality axioms in that signature.

The **derivable equalities and freshnesses** $\mathsf{U};\Delta \vdash_{\text{NEL}} r = s$ and $\mathsf{U};\Delta \vdash_{\text{NEL}} a\#r$ are generated by the rules in Figure 2.

$$\frac{}{\mathsf{U};\Delta \vdash_{\text{NEL}} r = r}\ (\mathbf{Refl}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NEL}} r = s \quad \mathsf{U};\Delta \vdash_{\text{NEL}} s = t}{\mathsf{U};\Delta \vdash_{\text{NEL}} r = t}\ (\mathbf{Trans}{=})$$

$$\frac{\mathsf{U};\Delta \vdash_{\text{NEL}} r = s}{\mathsf{U};\Delta \vdash_{\text{NEL}} s = r}\ (\mathbf{Symm}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NEL}} r = s \quad \mathsf{U};\Delta \vdash_{\text{NEL}} a\#r}{\mathsf{U};\Delta \vdash_{\text{NEL}} a\#s}\ (\mathbf{Trans}\#)$$

$$\frac{a\#X \in \Delta}{\mathsf{U};\Delta \vdash_{\text{NEL}} \pi(a)\#\pi{\cdot}X}\ (\#\mathbf{Ax}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NEL}} r_i = s_i \qquad (1 \le i \le n)}{\mathsf{U};\Delta \vdash_{\text{NEL}} \mathsf{f}(r_1,\ldots,r_n) = \mathsf{f}(s_1,\ldots,s_n)}\ (\mathbf{Cong})$$

$$\frac{\mathsf{U};\Delta, a\#X \vdash_{\text{NEL}} r = s \quad (a \notin atms(r,s))}{\mathsf{U};\Delta \vdash_{\text{NEL}} r = s}\ (\mathbf{Fresh}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NEL}} a\#r \quad \mathsf{U};\Delta \vdash_{\text{NEL}} b\#r}{\mathsf{U};\Delta \vdash_{\text{NEL}} (a\ b){\cdot}r = r}\ (\mathbf{Perm})$$

$$\frac{\mathsf{U};\Delta \vdash_{\text{NEL}} a\#r_i \quad (1 \le i \le n)}{\mathsf{U};\Delta \vdash_{\text{NEL}} a\#\mathsf{f}(r_1,\ldots,r_n)}\ (\#\mathbf{1}) \qquad \frac{}{\mathsf{U};\Delta \vdash_{\text{NEL}} a\#b}\ (\#\mathbf{2})$$

$$\frac{\mathsf{U};\Delta' \vdash_{\text{NEL}} a\#\theta(X) \text{ every } a\#X \in \Delta \quad ((\Delta \vdash r = s) \in \mathsf{U})}{\mathsf{U};\Delta' \vdash_{\text{NEL}} \pi{\cdot}(r\theta) = \pi{\cdot}(s\theta)}\ (\mathbf{Ax}_{\mathbf{\Delta \vdash r=s}})$$

$$\frac{\mathsf{U};\Delta' \vdash_{\text{NEL}} a\#\theta(X) \text{ every } a\#X \in \Delta \quad ((\Delta \vdash a\#r) \in \mathsf{U})}{\mathsf{U};\Delta' \vdash_{\text{NEL}} \pi(a)\#\pi{\cdot}(r\theta)}\ (\mathbf{Ax}_{\mathbf{\Delta \vdash a\#r}})$$

Figure 2: Derivable entailment in Nominal Equational Logic (NEL)

Remark 3.8. For the benefit of the interested reader, we compare the rules in Figure 2 and those in e.g. Figure 5.1 of [Clo09], where Nominal Equational Logic is presented in Clouston's thesis. This is not intended to be a formal proof of equivalence, but it should make 'obvious' that the we are talking about the same mathematical object, if differently presented.

- (**Refl**), (**Symm**), and (**Trans**$=$) and (**Trans**$\#$) in Figure 2 correspond to (REFL), (SYMM), and (TRANS) from [Clo09, Figure 5.1].
- (**Cong**), ($\#\mathbf{1}$), and ($\#\mathbf{2}$) taken together correspond to (SUBST), which is a single complex rule expressing intersubstitutability of derivably equal terms (the complexity of having three rules being offset by the complexity implicit in the arbitrary substitution used in (SUBST); for our purposes, the more 'expanded' version is much easier to prove things about).
- (WEAK) from Figure 5.1 is an admissible rule of Figure 2; we build weakening into ($\#\mathbf{Ax}$) here, whereas in Figure 5.1 it is an explicit rule plus a 'non-weakened' freshness axiom rule ($\not\#$-EQUIVAR).
- (ATM-INTRO) from Figure 5.1 has no direct counterpart in Figure 2; it is a product of the more structured sequent used in [Clo09] which combines equality with freshness judgements. Our design decision, to split the judgement-form from [Clo09] into separate freshness and equality judgements, seems to us the simpler and more elementary approach, but it is in any case purely a matter of presentation.
- (ATM-ELIM) from Figure 5.1 corresponds to (**Fresh**) in Figure 2; the side-condition $a\#(\nabla, \bar{a}, t, t')$ from (ATM-ELIM) corresponds the the side-condition $a \notin atms(r, s)$ in (**Fresh**).
  There is one slight difference, that (ATM-ELIM) requires you to add your freshness assumptions all at once, whereas (**Fresh**) allows you to add them one at a time. This is purely a design decision (cf. [GM09a, Remark 3.8]).
- ($\not\#$-EQUIVAR) is an admissible rule corresponding as discussed above to the specific design of ($\#\mathbf{Ax}$).
- (SUSP) from Figure 5.1 corresponds to (**Perm**) from Figure 2.
- The two axiom rules in Figure 2 are not explicitly written out in Figure 5.1, but are understood to be admitted 'as axioms'.

Remark 3.9. One difference between the NEL of Figure 2 and the NEL of Figure 5.1 in [Clo09] is that Clouston and Pitts admitted signatures of non-equivariant term-formers (so $\pi{\cdot}\mathsf{f}(r_1, \ldots, r_n) = (\pi{\cdot}\mathsf{f})(\pi{\cdot}r_1, \ldots, \pi{\cdot}r_n)$), whereas we do not and we keep signatures equivariant (so $\pi{\cdot}\mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}(\pi{\cdot}r_1, \ldots, \pi{\cdot}r_n)$).

So the NEL of this paper is a 'core NEL' which is actually a bit simpler than the NEL in the literature. What may surprise some readers is a point that appears to not be widely appreciated: we lose no expressivity in the simplification. See Subsections 6.3.2 and 6.3.3.[5]

Remark 3.10. We now compare Figures 1 and 2. The rules ($\mathbf{Ax}_{\mathbf{\Delta \vdash a\#r}}$) and (**Trans**$\#$) add axioms and equational reasoning for freshness, relative to the NA system from Figure 1. The rules ($\#\mathbf{1}$) and ($\#\mathbf{2}$) in Figure 2 also do not feature in Figure 1.

The rules ($\#\mathbf{1}$) and ($\#\mathbf{2}$) are needed here, and they are *not* part of some inductive definition e.g. like we saw in Definitions 3.2 or 3.4. Axioms can change what freshnesses are derivable; no axiom can change $fa_{\Delta}(r)$ or $atms(r)$.

So for instance, in conjunction with appropriate axioms, the NEL freshness judgement is undecidable. For instance we can axiomatise some computation using a term r that returns a if it terminates. Then $\vdash a\#r$ judges whether r terminates (if r does not terminate, then it is equal to the diverging function).

[5] A referee suggested viewing the NEL and NEL with equality only in this paper as two new systems, inspired by the NEL of [CP07] and the NEL with equality only of [Clo09]. These have similar properties; e.g. NEL has an explicit freshness judgement and is sound and complete with respect to both equality and freshness, but they leave out the orthogonal issue of non-equivariant function symbols. I agree, except that I do not think one should sell, or appear to sell, something as a new system, just for a relatively minor change of signature.

$$\frac{}{\mathsf{U};\Delta \vdash_{\text{NELeo}} r = r}\ (\mathbf{Refl}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NELeo}} r = s \quad \mathsf{U};\Delta \vdash_{\text{NELeo}} s = t}{\mathsf{U};\Delta \vdash_{\text{NELeo}} r = t}\ (\mathbf{Trans}{=})$$

$$\frac{\mathsf{U};\Delta \vdash_{\text{NELeo}} r = s}{\mathsf{U};\Delta \vdash_{\text{NELeo}} s = r}\ (\mathbf{Symm}) \qquad \frac{\mathsf{U};\Delta \vdash_{\text{NELeo}} r_i = s_i \qquad (1 \le i \le n)}{\mathsf{U};\Delta \vdash_{\text{NELeo}} \mathsf{f}(r_1,\ldots,r_n) = \mathsf{f}(s_1,\ldots,s_n)}\ (\mathbf{Cong})$$

$$\frac{\mathsf{U};\Delta, a\#X \vdash_{\text{NELeo}} r = s \quad (a \notin atms(r,s))}{\mathsf{U};\Delta \vdash_{\text{NELeo}} r = s}\ (\mathbf{Fresh})$$

$$\frac{(a,b \notin atms(r))}{\mathsf{U};\Delta, a\#vars(r), b\#vars(r) \vdash_{\text{NELeo}} (a\ b)\cdot r = r}\ (\mathbf{Perm})$$

$$\frac{\begin{array}{l}\mathsf{U};\Delta', b\#vars(r,s) \vdash_{\text{NELeo}} (b\ a)\cdot\theta(X) = \theta(X) \text{ every } a\#X \in \Delta \\ (\Delta \vdash r = s) \in \mathsf{U})\end{array}}{\mathsf{U};\Delta' \vdash_{\text{NELeo}} \pi\cdot(r\theta) = \pi\cdot(s\theta)}\ (\mathbf{Ax}_{\mathbf{\Delta\vdash r=s}})$$

Figure 3: Derivable entailment in Nominal Equational Logic with Equality Only (NELeo)

3.3. Nominal equational logic with equality only

We need just a little notation, which will also be useful later:

Notation 3.11. Write $\Delta, a\#vars(r)$ for $\Delta \cup \{a\#X \mid X \in vars(r)\}$.

Definition 3.12. A **nominal equational logic with equality only** theory U is a signature along with a set of equality axioms (no freshness axioms) in that signature.

The **derivable equalities** $\mathsf{U};\Delta \vdash_{\text{NELeo}} r = s$ are generated by the rules in Figure 3.

Remark 3.13. The presentation here is a rendering of Figure 1 from [Clo11]. Figure 1 in [Clo11] seems shorter than Figure 3 here, but its presentation uses plenty of macros and sugar. For instance, to unpack the notation used in (SUBST) from Figure 1 of [Clo11] requires all of page 5 from [Clo11]. The presentation here is more explicit, and also perhaps a little simpler too.

Remark 3.14. Comparing Figures 2 and 3, note the different treatment of the (**Perm**) rule, which in Figure 3 seems to have some overlap with (**Fresh**); similarly for the (**Ax**) rule. NEL with equality only is more profligate than NEL or NA with its generation of fresh resources.

3.4. Permissive-nominal algebra

Definition 3.15. Choose a fixed, arbitrary, and computable partition of $\mathbb{A}$ into two countably infinite halves $\mathbb{A}^<$ and $\mathbb{A}^>$, so that $\mathbb{A} = \mathbb{A}^< \uplus \mathbb{A}^>$.[6] We can do this since we assumed that $\mathbb{A}$ was countably infinite.

Definition 3.16. Define a notion of **permissive free atoms** of a nominal term $fa(r)$ as follows:

$$fa(a) = \{a\} \qquad fa(\pi\cdot X) = \pi\cdot\mathbb{A}^< \qquad fa(\mathsf{f}(r_1,\ldots,r_n)) = \bigcup_1^n fa(r_i)$$

[6] A fancy mathematician's name for this is *moiety*, meaning 'partition into two equal halves' (from the French *moitié*).

When this idea was first introduced to the nominal literature, the referees loathed it. They found a partition arbitrary. Where did it come from? How was it computed? Why do we need to make an arbitrary choice? This was eventually solved by taking the set of atoms to be defined as $\mathbb{A} = \mathbb{A}^< \uplus \mathbb{A}^>$, instead of taking the set of atoms and then partitioning it arbitrarily. If $\mathbb{A}$ had already been chosen, as is the case in this paper, then we specify that the moiety is *computable* and all is well.

Perhaps this is an an interesting litmus test to check whether you are talking to a mathematician or a computer scientist. Tell a mathematician 'partition this set' and he says 'OK'. Tell a computer scientist the same thing and he says 'How?'.

$$\frac{}{\mathsf{V} \vdash_{\text{PNA}} r = r}\ (\mathbf{Refl}) \qquad \frac{\mathsf{V} \vdash_{\text{PNA}} r = s \quad \mathsf{V} \vdash_{\text{PNA}} s = t}{\mathsf{V} \vdash_{\text{PNA}} r = t}\ (\mathbf{Trans})$$

$$\frac{\mathsf{V} \vdash_{\text{PNA}} r = s}{\mathsf{V} \vdash_{\text{PNA}} s = r}\ (\mathbf{Symm}) \qquad \frac{(a \notin fa(r),\ \ b \notin fa(r))}{\mathsf{V} \vdash_{\text{PNA}} (b\ a)\cdot r = r}\ (\mathbf{Perm})$$

$$\frac{\mathsf{V}; \Delta \vdash_{\text{PNA}} r_i = s_i \qquad (1 \leq i \leq n)}{\mathsf{V}; \Delta \vdash_{\text{PNA}} \mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}(s_1, \ldots, s_n)}\ (\mathbf{Cong}) \qquad \frac{((r{=}s) \in \mathsf{V})}{\mathsf{V} \vdash_{\text{PNA}} \pi\cdot(r\theta) = \pi\cdot(s\theta)}\ (\mathbf{Ax_{r=s}})$$

Figure 4: Derivable entailment in permissive-nominal Algebra (PNA)

Definition 3.17. A **permissive-nominal equality** E is a pair $l = r$.

A **permissive-nominal algebra theory** V is a signature along with a set of permissive-nominal equalities in that signature.

The **derivable equalities** $\mathsf{V} \vdash_{\text{PNA}} r = s$ are generated by the rules in Figure 4.

3.5. Expressivity

Atoms-as-constants and atoms inequality. One feature of nominal terms languages is that they treat atoms as a kind of bindable and permutable constant symbol. So note that distinct atoms a and b in the syntax actually denote distinct atoms in the denotation. Thus if ν is an atomic sort then we can axiomatise atoms-*in*equality as a binary term-former $\mathsf{neq} : (\nu, \nu)o$ with axioms

$$\vdash \mathsf{neq}(a, b) = \top \qquad \vdash \mathsf{neq}(a, a) = \bot$$

along with appropriate term-formers and equalities for $\top : o$ and $\bot : o$. If atoms behaved more like variables, so that 'a could be equal to b', then this would not make sense.

Non-equivariant term-formers can be emulated. Signatures are *equivariant* in that π commutes with term-formers; $\pi\cdot\mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}(\pi\cdot r_1, \ldots, \pi\cdot r_n)$ (Definition 2.9)—this is reflected in the models when in Definition 4.6 we take $\mathsf{f}^{\mathscr{I}}$ to be an equivariant function.

We can always pass term-formers atoms as arguments. For instance, abstraction over a base type τ can be axiomatised as a term-former $\mathsf{abs} : (\nu, \tau)\tau$ with a single axiom

$$b\#X \vdash \mathsf{abs}(a, X) = \mathsf{abs}(b, (b\ a)\cdot X).$$

So $\mathsf{abs}(a, \text{-})$ can be viewed as a 'non-equivariant' term-former. More on this in Subsections 6.3.2 and 6.3.3.

Example: a theory of functions. A theory of functions is given as follows (this example is modified from [GM10, Figures 1 and 4]). Assume a name sort ν, a base sort τ, and term-formers $\mathsf{var} : (\nu)\tau$, $\mathsf{app} : (\tau, \tau)\tau$, and $\mathsf{lam} : (\nu, \tau)\tau$. Sugar $\mathsf{var}(a)$ to a, $\mathsf{app}(r', r)$ to $r'r$, and $\mathsf{lam}(a, r)$ to $\lambda a.r$. Then axioms are:

$$\begin{array}{rlcrcl}
(\beta\mathbf{var}) & & \vdash & (\lambda a.a)X & = & X \\
(\beta\#) & b\#Z & \vdash & (\lambda b.Z)X & = & Z \\
(\beta\mathbf{app}) & & \vdash & (\lambda a.(Z'Z))X & = & ((\lambda a.Z')X)((\lambda a.Z)X) \\
(\beta\mathbf{abs}) & b\#X & \vdash & (\lambda a.(\lambda b.Z))X & = & \lambda b.((\lambda a.Z)X) \\
(\beta\mathbf{id}) & & \vdash & (\lambda a.Z)a & = & Z \\
(\alpha) & b\#Z & \vdash & \lambda b.(b\ a)\cdot Z & = & \lambda a.Z
\end{array}$$

Here is an example derivation, sketched out and written in natural deduction style to save space:

$$\dfrac{\dfrac{\dfrac{\dfrac{(a \notin vars_{\varnothing}(b))}{\lambda a.b = \lambda c.b}\,(\mathbf{Ax}_{\alpha})}{\lambda b.\lambda a.b = \lambda b.\lambda c.b} \quad \dfrac{}{a = a}\,(\mathbf{Refl})}{(\lambda b.(\lambda a.b))a = (\lambda b.(\lambda c.b))a} \quad \dfrac{\dfrac{(c \notin fa_{\varnothing}(a))}{(\lambda b.(\lambda c.b))a = \lambda c.((\lambda b.b)a)}\,(\mathbf{Ax}_{\beta\mathbf{abs}}) \quad \dfrac{\dfrac{}{(\lambda b.b)a = a}\,(\mathbf{Ax}_{\beta\mathbf{var}})}{\lambda c.((\lambda b.b)a) = \lambda c.a}\,(\mathbf{Tran})}{(\lambda b.(\lambda c.b))a = \lambda c.a}\,(\mathbf{Tran})}{(\lambda b.(\lambda a.b))a = \lambda c.a}\,(\mathbf{Tran}) \tag{1}$$

The corresponding theories and derivations in NEL would look much the same. In PNA we would replace a freshness side-condition like $b\#Z$ in $(\beta\#)$ with a concrete choice of $b \in \mathbb{A}^{>}$ (it does not matter which one). So we obtain the following PNA theory, where $b \in \mathbb{A}^{>}$ and $a \in \mathbb{A}^{<}$:

$$\begin{array}{rrcl}
(\beta\mathbf{var}) & (\lambda a.a)X & = & X \\
(\beta\#) & (\lambda b.Z)X & = & Z \\
(\beta\mathbf{app}) & (\lambda a.(Z'Z))X & = & ((\lambda a.Z')X)((\lambda a.Z)X) \\
(\beta\mathbf{abs}) & (\lambda a.(\lambda b.Z))X & = & \lambda b.((\lambda a.Z)X) \\
(\beta\mathbf{id}) & (\lambda a.Z)a & = & Z \\
(\alpha) & \lambda b.(b\ a)\cdot Z & = & \lambda a.Z
\end{array}$$

Undecidability of semantic freshness in general. Equality and semantic freshness can encode arbitrary complexity. For instance, consider two λ-terms r and s in the signature above (we do not even need unknowns). The question whether $\varnothing \vdash r = s$ is derivable, is undecidable. Even restricted forms of equality have this power. Consider a λ-term r which encodes some assertion and returns a if the answer is 'yes' and diverges if the answer is 'no'. Then just using the judgement $\varnothing \vdash (c\ a)\cdot r = r$ we can detect whether the problem encoded by r has answer 'yes' or 'no'; this equality test is *semantic freshness*. Thus, this restricted form of equality is also undecidable in general (more on semantic freshness later).

4. Denotations

We can give our logics a common denotation in nominal sets.

4.1. Definition of nominal sets

Definition 4.1. A **set with a permutation action** X is a pair $(|\mathsf{X}|, \cdot)$ of

- a **carrier set** $|\mathsf{X}|$ and
- a group action on the carrier set $(\mathbb{P} \times |\mathsf{X}|) \to |\mathsf{X}|$, written infix as $\pi\cdot x$.
  So, $id\cdot x = x$ and $\pi\cdot(\pi'\cdot x) = (\pi \circ \pi')\cdot x$ for every π and π' and every $x \in |\mathsf{X}|$.

Definition 4.2. Say $A \subseteq \mathbb{A}$ **supports** $x \in |\mathsf{X}|$ when for all permutations π, if $\pi(a) = a$ for all $a \in A$ then $\pi\cdot x = x$.

Definition 4.3. A **nominal set** is a set with a permutation action such that every element has a unique least finite supporting set $supp(x)$, called the **support** of x.[7]

X, Y will range over nominal sets.

Definition 4.4. Call a function from $|\mathsf{X}|$ to $|\mathsf{Y}|$ **equivariant** when $\pi\cdot f(x) = f(\pi\cdot x)$ for all $x \in |\mathsf{X}|$ and all permutations π.[8]

Lemma 4.5. *Given nominal sets X_i for $1 \leq i \leq n$ the following data defines a nominal set $\Pi_1^n\mathsf{X}_i$:*

$$|\Pi_1^n\mathsf{X}_i| = |\mathsf{X}_1| \times \cdots \times |\mathsf{X}_n| \qquad \pi\cdot(x_1, \ldots, x_n) = (\pi\cdot x_1, \ldots, \pi\cdot x_n)$$

[7] In fact, if a finite supporting set exists then so does a unique least such. See [GP01, Proposition 3.4], or [Gab11, Theorem 2.21].

[8] So a function is equivariant when it is completely symmetric with respect to the permutation action.

4.2. Denotation

Definition 4.6. A **model** $\mathscr{I}$ of a signature $\Sigma = (AtomicSort, BaseSort, TermFormer, ar)$ (Definition 2.2) consists of the following data:

- For each base sort τ a nominal set $[\![\tau]\!]^{\mathscr{I}}$.
- For each term-former f of arity $(\alpha_1, \ldots, \alpha_n)\tau$ an equivariant function $\mathsf{f}^{\mathscr{I}}$ from $\Pi_1^n [\![\alpha_i]\!]^{\mathscr{I}}$ to $[\![\tau]\!]^{\mathscr{I}}$, where $[\![\nu]\!]^{\mathscr{I}} = \mathbb{A}_\nu$.

Definition 4.7. A **valuation** ς to $\mathscr{I}$ assigns to each X an element $\varsigma(X) \in [\![sort(X)]\!]^{\mathscr{I}}$.

Call ς **permissive** when $supp(\varsigma(X)) \subseteq \mathbb{A}^<$ for all X.

Definition 4.8. Suppose $\mathscr{I}$ is a model and ς is a valuation to $\mathscr{I}$. Assign nominal terms denotations in $\mathscr{I}$ as follows:

$$[\![a]\!]^{\mathscr{I}}_\varsigma = a^{\mathscr{I}} \qquad [\![\pi{\cdot}X]\!]^{\mathscr{I}}_\varsigma = \pi{\cdot}\varsigma(X) \qquad [\![\mathsf{f}(r_1, \ldots, r_n)]\!]^{\mathscr{I}}_\varsigma = \mathsf{f}^{\mathscr{I}}([\![r_1]\!]^{\mathscr{I}}_\varsigma, \ldots, [\![r_n]\!]^{\mathscr{I}}_\varsigma)$$

Lemma 4.9. *Suppose $\mathscr{I}$ is a model, ς is a valuation to $\mathscr{I}$, r is a nominal term, and π is a permutation. Then $\pi{\cdot}[\![r]\!]^{\mathscr{I}}_\varsigma = [\![\pi{\cdot}r]\!]^{\mathscr{I}}_\varsigma$.*

If ς is a permissive valuation then $\pi{\cdot}[\![r]\!]^{\mathscr{I}}_\varsigma = [\![\pi{\cdot}r]\!]^{\mathscr{I}}_\varsigma$.[9]

Definition 4.10.

- Write $\mathsf{T}; \Delta \models_{\text{NA}} r = s$ when for every model $\mathscr{I}$ and valuation ς to $\mathscr{I}$, if $supp(\varsigma(X)) \subseteq fa_\Delta(X)$ for every X then $[\![r]\!]^{\mathscr{I}}_\varsigma = [\![s]\!]^{\mathscr{I}}_\varsigma$.
- Write $\mathsf{U}; \Delta \models_{\text{NEL}} r = s$ when for every model $\mathscr{I}$ and valuation ς to $\mathscr{I}$, if $supp(\varsigma(X)) \subseteq fa_\Delta(X)$ for every X then $[\![r]\!]^{\mathscr{I}}_\varsigma = [\![s]\!]^{\mathscr{I}}_\varsigma$ (cf. equation (38) and Definition 6.3 of [CP07]).
- Write $\mathsf{U}; \Delta \models_{\text{NEL}} a\#r$ when for every model $\mathscr{I}$ and valuation ς to $\mathscr{I}$, if $supp(\varsigma(X)) \subseteq fa_\Delta(X)$ for every X then $a \notin supp([\![r]\!]^{\mathscr{I}}_\varsigma)$ (cf. equation (38) and Definition 6.3 of [CP07]).
- Write $\mathsf{V} \models_{\text{PNA}} r = s$ when for every model $\mathscr{I}$ and permissive valuation ς to $\mathscr{I}$, $[\![r]\!]^{\mathscr{I}}_\varsigma = [\![s]\!]^{\mathscr{I}}_\varsigma$.

Soundness and completeness for NA.

Fix some model $\mathscr{I}$ and valuation ς to $\mathscr{I}$.

Lemma 4.11. *If $supp(\varsigma(X)) \subseteq fa_\Delta(X)$ for every X then $supp([\![r]\!]^{\mathscr{I}}_\varsigma) \subseteq fa_\Delta(r)$.*

Theorem 4.12. *$\mathsf{T}; \Delta \vdash_{NA} r = s$ if and only if $\mathsf{T}; \Delta \models_{NA} r = s$.*

Soundness and completeness for NEL.

Theorem 4.13.

- *$\mathsf{U}; \Delta \vdash_{NEL} r = s$ if and only if $\mathsf{U}; \Delta \models_{NEL} r = s$.*
- *$\mathsf{U}; \Delta \vdash_{NEL} a\#r$ if and only if $\mathsf{U}; \Delta \models_{NEL} a\#r$.*

Soundness and completeness for PNA.

Lemma 4.14. *If ς is permissive (Definition 4.7) then $supp([\![r]\!]^{\mathscr{I}}_\varsigma) \subseteq fa(r)$.*

Theorem 4.15. *$\mathsf{V} \vdash_{PNA} r = s$ if and only if $\mathsf{V} \models_{PNA} r = s$.*

The proofs of the soundness part of Theorems 4.12, 4.13, and 4.15 are easy and more-or-less by construction. The proofs of completeness are also easy by appropriate Herbrand constructions. In the literature, these results can be found as [GM09a, Theorems 4.24 and 4.39], [CP07, Theorems 7.4 and 10.10], and [Gab12b, Theorems 7.4.6 and 7.5.12].[10]

[9] Why the extra condition? Consider $a, a' \in \mathbb{A}^>$. If $a \in supp(\varsigma(X))$ and $a' \notin supp(\varsigma(X))$ then $[\![(a'\ a){\cdot}X]\!]^{\mathscr{I}}_\varsigma \neq [\![X]\!]^{\mathscr{I}}_\varsigma$, which would be unsound for the (**Perm**) rule of Figure 4.

Because a finite permutation can 'place' any finite number of atoms outside of $\mathbb{A}^<$, the restriction that $supp(\varsigma(X)) \subseteq \mathbb{A}^<$ does not affect expressivity. It is *not* the case that permissive-nominal algebra 'can only talk about atoms in $\mathbb{A}^<$'.

[10] The proof in [Gab12b] proved completeness with respect to permissive-nominal sets, which may have infinite support. It is easy to replay the proof for nominal sets models. A more abstract and interesting models-based approach is taken in a paper in preparation [Gab12a]. The result is the same, however we prove it.

5. Translation between NEL and NA

5.1. Expressing semantic freshness in (permissive-)nominal algebra

NA and syntactic freshness

'Free atoms of' is an intensional judgement on syntax (Lemma 5.1) and is not implied by 'not in the support of the denotation of' (Lemma 5.3)...

Lemma 5.1. $\mathsf{T}; \Delta \vdash_{NA} r = s$ *does not imply that* $a \in fa_\Delta(r)$ *if and only if* $a \in fa_\Delta(s)$.

Proof. Consider a theory with one name sort ν, one base sort τ, one term-former f with arity $(\nu)\tau$, and one axiom $\varnothing \vdash \mathsf{f}(a) = \mathsf{f}(b)$. Then $a \in fa_\varnothing(\mathsf{f}(a))$ and $a \notin fa_\varnothing(\mathsf{f}(b))$. □

Definition 5.2. Write $\mathsf{T}; \Delta \models_{\text{NA}} a\#r$ when for every model $\mathscr{I}$ and valuation ς to $\mathscr{I}$, if $supp(\varsigma(X)) \subseteq fa_\Delta(X)$ for every X, then $a \notin supp([\![r]\!]^{\mathscr{I}}_\varsigma)$.

Lemma 5.3. *It is not necessarily the case that* $\mathsf{T}; \Delta \models_{NA} a\#r$ *implies* $a \notin fa_\Delta(r)$.

Proof. Using the previous example, it is not hard to verify that $\mathsf{T}; \Delta \models_{\text{NA}} a\#\mathsf{f}(a)$, yet $a \in fa_\varnothing(\mathsf{f}(a))$. □

NA and semantic freshness

...however '$(b\ a)\cdot x = x$ for fresh b' is an extensional judgement (Lemma 5.4; this is the real meaning of the rule identified in Section 11 of [CP07]) and is implied by 'not in the support of the denotation of' (Corollary 5.5).

Recall the notation $\Delta, a\#vars(r)$ from Notation 3.11.

Lemma 5.4. *Suppose* $\mathsf{T}; \Delta, b\#vars(r) \vdash_{NA} (b\ a)\cdot r = r$ *and* $\mathsf{T}; \Delta \vdash_{NA} r = s$.
Then $\mathsf{T}; \Delta, b\#vars(s) \vdash_{NA} (b\ a)\cdot s = s$.

Proof. By routine calculations using Theorems 3.2.2 and 3.2.3 of [GM09a]. □

Corollary 5.5. $\mathsf{T}; \Delta \models_{NA} a\#r$ *if and only if* $\mathsf{T}; \Delta, b\#vars(r) \vdash_{NA} (b\ a)\cdot r = r$.

Proof. Write $\Delta' = \Delta, a\#vars(r)$ and choose some fresh b (so $b \notin fa_{\Delta'}(r)$).

Suppose $\mathsf{T}; \Delta \models_{\text{NA}} a\#r$. By assumption if $\mathscr{I}$ is a model and ς is such that $supp(\varsigma(X)) \subseteq fa_{\Delta'}(X)$ for every $a\#X \in \Delta'$, then $a \notin supp([\![r]\!]^{\mathscr{I}}_\varsigma)$.

By assumption $b \notin fa_{\Delta'}(r)$ so by Lemma 4.11 $b \notin supp([\![r]\!]^{\mathscr{I}}_\varsigma)$. It follows by Definition 4.2 that $(b\ a)\cdot[\![r]\!]^{\mathscr{I}}_\varsigma = [\![r]\!]^{\mathscr{I}}_\varsigma$. By Lemma 4.9 $[\![(b\ a)\cdot r]\!]^{\mathscr{I}}_\varsigma = [\![r]\!]^{\mathscr{I}}_\varsigma$. By Completeness (the right-to-left part of Theorem 4.12) $\mathsf{T}; \Delta' \vdash_{\text{NA}} (b\ a)\cdot r = r$.

The converse implication follows similarly, using Soundness (the left-to-right part of Theorem 4.12). □

Corollary 5.5 has appeared (at least) as Theorem 5.5 from [GM07], Lemma 4.51 from [GM09a], and also as Theorem 5.5.7 of [Clo09].

A corresponding result holds for PNA, and is even simpler to state:

Definition 5.6. Write $\mathsf{V} \models_{\text{PNA}} a\#r$ when for every model $\mathscr{I}$ and permissive valuation ς to $\mathscr{I}$, $a \notin supp([\![r]\!]^{\mathscr{I}}_\varsigma)$.

Corollary 5.7. $\mathsf{V} \models_{PNA} a\#r$ *if and only if* $\mathsf{V} \vdash_{PNA} (b\ a)\cdot r = r$, *where we choose* $b \notin fa(r)$.

5.2. Models

Definition 5.8. Given an NEL theory U create a new theory U' with the same signature such that

- An axiom $\Delta \vdash r = s$ maps to itself.
- An axiom $A = \Delta \vdash a\#r$ maps to $A' = \Delta, b\#X_1, \ldots, b\#X_n \vdash (b\,a)\cdot r = r$ where $\{X_1, \ldots, X_n\}$ is the set of unknowns appearing in the axiom and b is fresh.

Theorem 5.9. *The class of models defined by* U *in NEL is equal to the class of models defined by* U' *in NA.*

Proof. Using Corollary 5.5. For more details see [GM09a, Section 5]. □

So by Theorem 5.9 for every derivable NEL freshness judgement there is a corresponding derivable NA equality judgement, in a corresponding theory, which has the same meaning in the models.

Conversely, any NA theory is already an NEL theory, and it is not hard to prove the following adjoint to Theorem 5.9:

Theorem 5.10. *The class of models defined by* U *in NA is equal to the class of models defined by* U *considered as an NEL theory.*

5.3. Translation of derivations

The arguments above are not constructive; they do not show how individual derivations might translate. We now indicate how to do this.

The idea is that NEL semantic freshness judgements transform to NA equality judgements of the form $(b\,a)\cdot r = r$ for fresh b.

Consider some NEL derivation Π of $\mathsf{U}'; \Delta \vdash r = s$ or $\mathsf{U}'; \Delta \vdash a\#r$. The challenge is to translate the freshness part. First, we note down *all* the unknowns used in Π; write these $X_1, \ldots, X_n$. We choose a fresh b that does not appear anywhere in Π and write Δ' for the freshness context which is Δ augmented with $b\#X_1, \ldots, b\#X_n$.

An instance of $(\mathbf{Ax}_{\mathbf{\Delta \vdash a\#r}})$ with π and θ translates to the following derivation-fragment:

$$\frac{}{\mathsf{U}'; \Delta' \vdash (\pi \circ (b\,a))\cdot(r\theta) = \pi\cdot(r\theta)}\,(\mathbf{Ax}_{\mathbf{A}'})$$

Lemma 5.11. *If* $\mathsf{U}; \Delta \vdash_{NA} r = s$ *then* $\mathsf{U}; \Delta \vdash_{NA} \pi\cdot r = \pi\cdot s$.

Proof. By a routine induction on derivations. See [GM09a, Theorem 3.2.2]. □

An instance of $(\mathbf{Trans}\#)$ translates to the following derivation-fragment, which for clarity we represent schematically; we use Lemma 5.11 and rules $(\mathbf{Symm})$, $(\mathbf{Trans})$, and so on:

$$\frac{(b\,a)\cdot s = (b\,a)\cdot r \quad (b\,a)\cdot r = r \quad r = s}{(b\,a)\cdot s = s}$$

The rules $(\#\mathbf{1})$ and $(\#\mathbf{2})$ easily translate; we consider only the case of $(\#\mathbf{1})$. This uses the fact that $(b\,a)\cdot \mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}((b\,a)\cdot r_1, \ldots, (b\,a)\cdot r_1)$; again we indicate it schematically:

$$\frac{(b\,a)\cdot r_i = r_i \quad (1 \leq i \leq n)}{(b\,a)\cdot \mathsf{f}(r_1, \ldots, r_n) = \mathsf{f}(r_1, \ldots, r_n)}\,(\mathbf{Cong})$$

What we now have is:

- From an NEL derivation of $\mathsf{U}; \Delta \vdash r = s$ an NA derivation of $\mathsf{U}'; \Delta' \vdash r = s$, from which we can obtain a derivation of $\mathsf{U}'; \Delta \vdash r = s$, using $(\mathbf{Fresh})$ repeatedly.
- From an NEL derivation of $\mathsf{U}; \Delta \vdash a\#r$ an NA derivation of $\mathsf{U}'; \Delta' \vdash (b\,a)\cdot r = r$.

Translating an NA derivation to an NEL derivation is no harder. An NA derivation is almost an NEL derivation already, except most notably for the different treatment of freshness assumptions in the axiom rule $(\mathbf{Ax})$. We just use the fact that $(\#\mathbf{1})$ and $(\#\mathbf{2})$ can emulate the inductive definition of $fa_\Delta(r)$.

6. NA, NEL, and syntactic/semantic freshness

6.1. Notation for semantic freshness

We wrote $\Delta \vdash a\#r$ in [GM07] where here we write $a \notin fa_{\Delta}(r)$. The notation here is designed to make it impossible to misread the side-condition $a \notin fa_{\Delta}(r)$ as a logical judgement, as can happen to a reader exposed to the comments on nominal algebra in [CP07] and [Clo11] without further context.

Where did this notation come from? The notation $\Delta \vdash a\#r$ was developed with Urban and Pitts in [UPG04] and is the established notation for nominal terms, used e.g. in nominal rewriting and αProlog [FG07, CU08]. Its use continues; see e.g. [CF08a, LV11], which are examples of quite a large literature.

When one considers equational reasoning from axioms, there is a design choice: should derivable freshness take the equations in the theory into account? If so, then we also need explicit rules for reasoning about freshness from axioms that generalise the Urban-Pitts-Gabbay style freshness judgement (and ought to be sound and complete for it); if not, we do not need such rules. Either design choice is justifiable as a generalisation from the initial theory and notation of [UPG04]; however, in the case of NA the syntactic freshness reasoning (by design) no longer matches the semantics. In this sense, the $\Delta \vdash a\#r$ notation can be misleading and we have replaced it in this paper.

The choice is superficial, in the sense that one can do without freshness axioms and judgements (NA). Thus, if we want to prove something about NEL or NEL with equality only, we might be able to prove it about NA or PNA (which are smaller systems) and lift to NEL by translation.

Semantic and syntactic freshness had already parted company with the introduction of nominal rewriting [FGM04, FG07], where the interaction of syntactic freshness with rewrites (directed equality) is a topic of great importance, notably for confluence properties. In particular we have studied *closed rules* [FGM04] and *uniform rules* [FGM04, FG07], where the interest is specifically how rewrites can influence the free atoms of a term.

We do have the following result, which is not hard to prove:

Lemma 6.1. *$a \notin fa_{\Delta}(r)$ if and only if $a \notin supp([\![r]\!]^{\mathscr{I}}_{\varsigma})$, for* every *model $\mathscr{I}$ and valuation ς to that model.*

Proof. It suffices to construct a Herbrand model out of closed syntax quotiented by α-equivalence, possibly adding term-formers to ensure that we have enough closed terms. □

In addition, in permissive-nominal terms quotiented by α-equivalence, $a \notin fa(r)$ if and only if a is fresh for r in permissive-nominal sets. That is off-topic for this paper; see [Gab12b] for more details.

6.2. Three positions of Clouston and Pitts

Three positions are taken regarding NEL and NA in [CP07] (which introduced NEL), [Clo09] (Clouston's PhD thesis), and [Clo11] (which is based on the second half of [Clo09]):

- Position 1. *"NA does not provide a complete axiomatisation of the semantic notion of [nominal] freshness within nominal sets."* [CP07, Section 11]
- Position 2. *"... freshness can be expressed in terms of equality, [so the difference between NA and NEL] amounts to different design choices, rather than any deeper distinction [but] NEL's choice is clearly ... appropriate ... for approaching freshness as a first-class subject of study."* [Clo09, Section 8.2]
- Position 3. Semantic freshness is dropped from nominal equational logic. It is approached using equality alone [Clo11, Figure 1].

In the light of the maths we have seen so far we can note the following:

1. Nominal algebra provides a complete notion of semantic freshness. This is built in to the design and is noted explicitly in Theorem 5.5 from [GM07], and also by Theorem 5.5.7 of [Clo09] and by the use of NEL with equality only in the second half of [Clo09] and [Clo11]. As mentioned in Subsection 6.1 the notation $a \notin fa_{\Delta}(r)$ is designed to make the confusion between a side-condition and a logical judgement impossible.
   Sadly this has more than historical interest, and confusion is still propagated in the literature today. Most recently in [Clo11, Section 7], Clouston writes *"freshness in NA is sound, but not complete, for freshness in the underlying nominal sets interpretation"*.
   We hope that by now, it is clear that this sentence is meaningless and rests on a category error.
2. Semantic freshness is a definitional extension of NA. Contrary to the quote given above, an explicit semantic freshness judgement is not clearly appropriate.[11] Nominal Lawvere Theories use as syntax *NEL with equality only* and the approach to freshness uses equality only, after the manner of NA.
3. The semantic freshness judgement was eventually dropped, shifting from NEL to NEL with equality only.
   The logic of [Clo11] is called 'nominal equational logic'. It should not be, since it is not the same as the logic of the same name from [CP07]. We follow the terminology of [Clo09] and call it *NEL with equality only*.

These logics are laid out for the reader to inspect and compare, in a uniform presentation and for the first time together, in Figures 1, 2, and 3.

Remark 6.2. NEL and NEL with equality only are not truly equational. They are Horn clause logics, in which propositions to the left of a turnstile are restricted to have (intuitively) the meaning '$(b\ a){\cdot}x = x$ for fresh b'. These judgements are expressive; e.g. in the presence of the right axioms semantic freshness can be undecidable as we saw in Subsection 3.5. Thus Clouston wrote *"freshness is a more complicated concept that must be defined"* [Clo09, Section 5.5].

Remark 6.3. It may surprise the reader that the logic in [Clo11] does use a syntactic freshness side-condition after all. It is just a little hidden by the presentation: in our notation it is $a \notin atms(r)$.

We trace this back to [Clo11]. In (**Subst**) in Figure 5.4 of [Clo11] there is a simultaneous choice of fresh atoms $\bar{a}_i$; this notation is explained in an unnumbered definition just before Remark 3.2 of that paper. Unpacking what this in syntax-directed terms reveals a freshness judgement written $a\#r$ which is not written out in full, but which corresponds to $a \notin atms(r)$ from Definition 3.4.

Remark 6.4 ($a \notin atms(r)$ *and* $a \notin fa_{\Delta}(r)$ *'morally' equal*)**.** The difference between $a \notin atms(r)$ and $a \notin fa_{\Delta}(r)$ is to some extent an artefact. In the permissive-nominal terms framework of [Gab12b] $atms(r)$ and $fa(r)$ become equal. The inequality between $atms$ and fa here arises because our representation of syntax is more concrete than it absolutely needs to be; our syntax 'remembers' names of bound atoms, in the sense that $[a]a$ and $[b]b$ are distinct formal syntax, whereas in permissive-nominal syntax they can be taken to be identical (just as we do for normal first-order syntax).

There is a real sense in which NEL with equality only uses the same syntactic freshness judgement as NA. They look different only because of how they are projected to concrete syntax. We predict that this point will become clearer with time as the literature matures.

Remark 6.5. The computational content of $fa_{\Delta}(r)$ also exists in the NEL family, in a certain sense.

Broadly speaking: where NA uses $a \notin fa_{\Delta}(r)$; NEL with equality uses $a \notin atms(r)$ to choose a fresh atom, extends the freshness context with $a\#vars(r)$, and then produces an equality $(b\,a){\cdot}r = r$, which we recognise as the equality of Corollary 5.5.

Solving this equality triggers a cascade of reasoning which follows the syntax-directed definition of $fa_{\Delta}(r)$.

[11] In the theory. As usual, in an implementation we should put in whatever users want and, presumably, compile the richer language down to a core system.

The reader can see this happen in (**Perm**) and (**Ax**) in Figure 3. Returning to [Clo11], choices of atoms $\bar{a}_i$ disjoint from $atms(r)$, written $\bar{a}_i\#r$ in [Clo11], are used to generate an equality assertion which is written out in full in [Clo11, Equation (9)].

One might debate which of these strategies is more computationally efficient, but in any case NEL and NEL with equality only *do* have a syntactic freshness judgement, accompanied by machinery which is at least as complicated as $fa_{\Delta}(r)$.

Remark 6.6 (*И at the meta-level*)**.** We can read '$a \notin atms(r)$' as 'a is fresh'. But here is an interesting alternative reading: one can argue that what is really happening in (**Fresh**) in Figures 1 and 2, and in (**Fresh**), (**Perm**) and (**Ax**) in Figure 3 (and in (**Subst**) in Figure 5.4 of [Clo11]), is the Gabbay-Pitts И quantifier. By that reading, syntactic freshness is simulated by swapping plus an incognito meta-level new-quantifier.

6.3. Further comments

6.3.1. On the usefulness and necessity of reinterpretation

The NEL of this paper is a reinterpretation of Clouston and Pitts's work. The presentation of NEL used a different syntax from NA. Why reinterpret the syntax in this paper?

NEL uses a 'cylindric' style syntax (in the sense of *cylindric algebra* [HMT85]). See Definition 4.2 of [CP07] for their syntax, and Figure 3 of [CP07] for their permutation and substitution actions. Compare with Definitions 2.5, 2.9, and 2.10 here.

Unfortunately, this choice of 'non-standard syntax' breaks compatibility with the large body of work on nominal terms, and makes it difficult to compare NEL work with the quite substantial body of work using nominal terms. One useful product of this paper is a more explicit presentation of NEL which bridges the gap between these two bodies of literature.[12]

Providing an inherently finitely presentable NEL syntax is a good idea. There is nothing wrong with a more abstract presentation, but a finite presentation of the syntax should be provided too, as a service to and courtesy to the rest of the literature. This paper plugs that gap and uses the syntax to clearly compare the members of the (P)NA/NEL family, arguably for the first time.

There is also a little more. The cylindric style is that NEL almost always requires infinite signatures; e.g. any non-equivariance in the signature will immediately cause this, because there are infinitely many atoms to consider. For *implementation*, some explicitly finitely presentable syntax for NEL is not only good practice but a necessity: users input ASCII text, not infinite sets. Furthermore, work on complexity like [CF08b, LV10], depend on the finite presentability.

And finally, lest we forget, implementability was one of the motivations for nominal techniques in the first place. In principle, variables are optional and we could use combinators, but real implementations use variables and it was understanding these variables that motivated the development of nominal sets in the first place.

6.3.2. Non-equivariant term-formers

We continue the discussion of Remark 3.9. The syntax used in [CP07] uses non-equivariant term-formers.

Does this actually matter? Not for the logic. We explain how, in a moment.

But first, suppose we want non-equivariant term-formers in the logics of this paper, after all. How would we go about it?

- We can simply extend nominal terms with non-equivariant term-formers, and thus allow terms of the form $(\pi\cdot f)(r_1,\ldots,r_n)$ alongside $\pi\cdot X$ in Definition 2.5. We would also need to take $\pi\cdot(\mathsf{f}(r_1,\ldots,r_n)) = (\pi\cdot\mathsf{f})(\pi\cdot r_1,\ldots,\pi\cdot r_n)$ in Definition 2.9. This possibility was discussed right back during the original design of nominal terms.

[12] We do not consider the non-equivariant sorts used in some later NEL work. Typing systems for nominal terms remain an open topic.

- We can admit a class of non-equivariant constant symbols, so even if term-formers are equivariant, some constant symbols might not be. This is the approach taken in the PNL of [Gab12b], where in addition support can be infinite.

In fact this does not matter for expressivity in an equational logic. Non-equivariant operation symbols can map to equivariant term-formers restricted to apply to a finite list of distinct atoms. A term-former $\mathsf{f}(\text{-})$ with support $\{a_1, \ldots, a_n\}$ maps to an equivariant term-former with arguments $\mathsf{f}'(a_1, \ldots, a_n, \text{-})$ (we discuss 'junk' in the next subsection). Complexity is migrated between the sort system and the notion of term-former/operator (cf. the simple examples of neq and abs from Subsection 3.5).

If the permutation action satisfies non-trivial equalities, then these are converted to axioms in the obvious way. For instance, if we want a sort τ and constant symbols of sort τ that are isomorphic to unordered pairs of atoms from $\mathbb{A}_\nu$, then the translation would require a term-former $\mathsf{f} : (\nu, \nu)\tau$ along with a single axiom $\mathsf{f}(a, b) = \mathsf{f}(b, a)$.

How this generalises to an arbitrary nominal set follows from a general result, Theorem 3.12 of [Gab09b], which describes how every nominal set can be expressed as a disjoint sets union of orbits under the permutation action, and equalities within each orbit can be captured by a finite set of axioms.[13] A translation of a non-equivariant signature to an equivariant signature plus for each orbit a finite set of axioms, follows immediately. A translation of derivations between the system with non-equivariant term-formers and the system with axioms for term-formers applied to distinct atoms, is rather easy to construct (and resembles the more complex translation to eliminate 'junk' in the proof of Theorem 6.9).

Note that the axiomatic theory hidden in the permutation action can be non-trivial. Consider a nominal set which for every program P and pair of atoms a and b contains an element $P_{a,b}$ such that $P_{a,b} = P_{b,a}$ if and only if P halts.

6.3.3. Junk in models

Continuing the previous point, if we emulate 'a constant $\mathsf{C}_{a,b}$ with support $\{a, b\}$' by taking a term-former C taking two atoms as arguments and writing $\mathsf{C}(a, b)$, then we also admit the junk term $\mathsf{C}(a, a)$. It is not possible, with an equivariant signature, to consider the term $\mathsf{C}(a, b)$ alone.

So in the translation from the NEL of [CP07] to the 'NEL' of this paper, some 'junk' is added to the models. Does this make a difference? Not to the logic: the same non-junk equalities are derivable either way. We show how this works for NA:

Definition 6.7. Fix a signature Σ (Definition 2.2) and pick out some set of term-formers $\mathsf{f}_i : (\nu_{i1}, \ldots, \nu_{in_i}, \alpha_{i1}, \ldots, \alpha_{im_i})\tau$ for $i \in I$; these are 'pretending' to be non-equivariant by taking atoms as arguments.

Say a term of the form $\mathsf{f}_i(s_1, \ldots, s_{n_i}, r_1, \ldots, r_{m_i})$ **is junk** when the set $\{s_1, \ldots, s_{n_i}\}$ does not consist of n_i distinct atoms (in the example above $\mathsf{C}(a, a)$ is junk and $\mathsf{C}(a, b)$ is not junk).

If a term r contains a junk subterm then say that r **contains junk** (so every term that is junk, contains junk, but not every term that contains junk, is junk). Say a theory T **contains junk** when it contains an axiom that contains junk. Finally, if a derivation contains a junk term then say the derivation **contains junk**.

We will prove a conservative extension result for the system in which junk is allowed, with respect to the system in which no junk is allowed; this is Theorem 6.9. First, we need a key lemma about nominal algebra:

Lemma 6.8. *If* $\mathsf{T}; \Delta \vdash_{NA} r = s$ *and* $r : \nu$ *and* $s : \nu$ *then precisely one of the following holds:*

- $r = \pi{\cdot}X$ *and* $s = \pi'{\cdot}X$ *and* $a\#X \in \Delta$ *for every* a *such that* $\pi(a) \neq \pi'(a)$.
- $r = a$ *and* $s = a$ *for some* $a \in \mathbb{A}_\nu$.

[13] Any set with a group action can be so partitioned. And why finite? Because support is finite; we choose a representative x of the orbit with support $supp(x) = \{a_1, \ldots, a_n\}$ and write an axiom $\pi{\cdot}\mathsf{f}(a_1, \ldots, a_n) = \mathsf{f}(a_1, \ldots, a_n)$ for any of the π such that $\pi{\cdot}x = x$. An unsophisticated calculation gives an upper bound of $n!$ distinct axioms.

Proof. By soundness and completeness of NA with respect to its models (Theorem 4.12), in which $[\![\nu]\!]^{\mathscr{I}} = \mathbb{A}_\nu$ (Definition 4.6).[14] □

Theorem 6.9. *Suppose* T, r, *and* s *do not contain junk. Then* $\mathsf{T}; \Delta \vdash_{\text{NA}} r = s$ *is derivable if and only if* $\mathsf{T}; \Delta \vdash_{\text{NA}} r = s$ *has a derivation that does not contain junk.*

Proof. Clearly if $\mathsf{T}; \Delta \vdash_{\text{NA}} r = s$ has a derivation Π that does not contain junk then $\mathsf{T}; \Delta \vdash_{\text{NA}} r = s$ has a derivation.

Suppose we have a derivation Π of $\mathsf{T}; \Delta \vdash_{\text{NA}} r = s$, and suppose Π contains junk. Note that T contains no junk, and neither does $r = s$. We now perform a transformation on Π specified as follows:

- For each of the designated term-formers f_i choose a completely fresh set of atoms $a_{i1}, \ldots, a_{in_i}$ (so these atoms do not occur anywhere in Π).
- Define a translation $(\text{-})^*$ by $a^* = a$, $(\pi \cdot X)^* = \pi \cdot X$, $\mathsf{f}(r_1, \ldots, r_n)^* = \mathsf{f}(r_1^*, \ldots, r_n^*)$ if $\mathsf{f}(r_1, \ldots, r_n)$ is not junk,[15] and $\mathsf{f}_i(s_1, \ldots, s_{n_i}, r_1, \ldots, r_{m_i})^* = \mathsf{f}_i(a_{i1}, \ldots, a_{in_i}, r_1^*, \ldots, r_{m_i}^*)$ otherwise.
- Finally, translate Π by replacing r with r^* and replacing any instance of $(\mathbf{Cong})$ whose conclusion is junk, as follows:

$$\dfrac{\begin{array}{ccccc}\Pi_1 & & \Pi_{n_i} & & \Pi_{m_i} \\ \vdots & & \vdots & & \vdots \\ s_1 = s_1' & \ldots & s_{n_i} = s_{n_i}' & \ldots & r_{m_i} = r_{m_i}'\end{array}}{\mathsf{f}_i(s_1, \ldots, s_{n_i}, r_1, \ldots, r_{m_i}) = \mathsf{f}_i(s_1', \ldots, s_{n_i}', r_1', \ldots, r_{m_i}')}\ (\mathbf{Cong})$$

is replaced by

$$\dfrac{\dfrac{}{a_{i1} = a_{i1}}\ (\mathbf{Refl}) \quad \ldots \quad \dfrac{}{a_{in_i} = a_{in_i}'}\ (\mathbf{Refl}) \quad \ldots \quad \begin{array}{c}\Pi_{m_i} \\ \vdots \\ r_{m_i}^* = (r_{m_i}')^*\end{array}}{\mathsf{f}_i(a_{i1}, \ldots, a_{in_i}, r_1^*, \ldots, r_{m_i}^*) = \mathsf{f}_i(a_{i1}, \ldots, a_{in_i}, (r_1')^*, \ldots, (r_{m_i}')^*)}\ (\mathbf{Cong}).$$

Note that if r contains no junk then $r^* = r$, so this does not affect the conclusion of Π. Note also that Lemma 6.8 implies that we do not have to consider the possibility in the replacement above that $\mathsf{f}_i(s_1, \ldots, s_{n_i}, r_1, \ldots, r_{m_i})$ is junk and $\mathsf{f}_i(s_1', \ldots, s_{n_i}', r_1', \ldots, r_{m_i}')$ is not.

It is not hard to verify that this transformed labelled tree is a derivation. We consider the two interesting cases:

- For an instance of $(\mathbf{Perm})$ we note that r' contains junk if and only if $(a\ b) \cdot r'$ does, and furthermore the junk is in the same parts of the term, and because we chose our atoms $\{a_{i1}, \ldots, a_{in_i}\}$ fresh, $(a\ b)$ does not affect them. It follows that $(a\ b) \cdot (r^*) = ((a\ b) \cdot r)^*$. Also, again because we chose our atoms fresh, it follows that $a, b \notin r^*$.
- For an instance of $(\mathbf{Ax}_{\mathbf{\Delta} \vdash \mathbf{r'} = \mathbf{s'}})$ we change the relevant θ to θ^* which maps X to $\theta(X)^*$. By assumption r' and s' in the axiom do not contain junk, and this means in particular that each occurrence of f_i in r' and s' is not at the head of a junk term and so occurs applied to n_i distinct atoms. It follows that $(r'\theta^*) = (r'\theta)^*$ and $(s'\theta^*) = (s'\theta)^*$. Also, since the atoms a_{ij} are chosen fresh θ^* satisfies all the necessary freshness conditions.

The rest is routine. □

[14] The important point here is that the interpretation of ν is *always* $\mathbb{A}_\nu$, and $[\![a]\!]_\zeta^{\mathscr{I}}$ is *always* equal to a, not matter what the model and what axioms are in T.

[15] It might still *contain* junk, but that is handled recursively by $(\text{-})^*$.

Junk is a common phenomenon. Models of Peano arithmetic admit non-standard elements which cannot be detected from within the Peano theory itself, cf. also the upward Löwenheim-Skolem theorem [Hod93]. Similarly models of Higher-Order Logic come with function-spaces of various sizes.

Also similarly, by dropping atoms-abstraction $[a]r$ from nominal terms we lost some control over models with respect to the NA of [GM09a]. This is discussed in [GM09a], and NEL did the same in [Clo10]. Though we can say in axioms 'a is abstracted in $\mathsf{abs}(a,r)$' (cf. Subsection 3.5) it is not possible (without negation or implication) to insist '*and* $[\![\mathsf{abs}(a,r)]\!]^{\not\zeta}_{\zeta}$ *is exactly equal* to $[a][\![r]\!]^{\not\zeta}_{\zeta}$'.

There is nothing wrong with non-equivariant term-formers (they give us extra models) but for designing a core equational reasoning system we can take them or leave them (the extra models do not affect the entailment relation). For more expressive logics, this can matter more, and in fact things can get quite subtle; see for instance [Gab12b, Gab12a].

6.3.4. The judgement form

The NEL of [CP07] has just one judgement form, written $\nabla \vdash \bar{a}\#t{\approx}t' : \mathsf{s}$. See equation (33) in Section 6 of [CP07]. Here ∇ is a freshness context, $\bar{a}$ is a finite set of atoms, t and t' are terms, and s is a sort.

This difference does not matter. In this paper, that single judgement would be represented as $\nabla \vdash a\#t$ for each a in $\bar{a}$, and further $\nabla \vdash t = t'$.

6.3.5. Natural deduction presentation

Early presentations of nominal algebra used natural deduction style derivations, rather than the sequent presentation used in later publications and here, see e.g. [Gab05]. When writing out examples, the natural deduction style can be a lot more compact and easier to typeset; we used it in this paper in equation 1 in Subsection 3.5.

This is purely a matter of presentation.

7. PNA versus NA

Permissive-nominal techniques were introduced in [DGM09, DGM10]. PNA is genuinely different from NA and NEL (with or without equality) in two respects:

- PNA has a slightly but significantly different proof-theory because Figure 4 has no (**Fresh**) rule.
  We could also lose the (**Perm**) by a quotient of syntax; see the next point.
- The permissive-nominal treatment of nominal terms does not involve a mutable freshness context Δ (this is related to the previous point). α-equivalence can be taken as 'just quotient by α-equivalence', like in traditional syntax. The interested reader is referred to [Gab12b].

PNA is part of a broader 'permissive' programme by the author with Mulligan and Dowek. We have only presented here a simplified fragment of the more general theory surveyed in [Gab12b], applied specifically to equational reasoning.

Intuitively, permissive-nominal techniques apply (**Fresh**) infinitely many times and fix the resulting freshness context. So we can think of permissive-nominal reasoning as being nominal reasoning in an infinite freshness context in which $a\#X$ for every $a \in \mathbb{A}^{>}$ and every X.

Comparing Figures 2, 3, 1, and 4, we see a decrease in complexity; Figure 4 differs from normal first-order algebra only in adding the π in the axiom rule. In permissive-nominal techniques, names and binding become structural properties of syntax instead of the quasi-logical—though still syntax-directed—definitions-in-context that they have on nominal terms. This means in plain English that we can 'just quotient' by α-equivalence just as we do in first- and higher-order syntax, and we do not have to worry about the non-structural (**Fresh**) rule since this is replaced by the use of $\mathbb{A}^{<}$ and $\mathbb{A}^{>}$.

This is currently the author's preferred technology for nominal reasoning. Empirically, we seem to get a clearer view of the underlying mathematics.

The permissive-nominal syntax of this paper is simplified. In other work, e.g. [Gab12b], we allow X with *permission sets* which are sets of the form $\pi{\cdot}\mathbb{A}^<$.[16] In effect, this paper considers a syntax in which all X have permission set $\mathbb{A}^<$. This does not affect expressivity because we can emulate unknowns with different permission sets by writing $\pi{\cdot}X$.

Remark 7.1. The reader who wants to see how PNA reasoning maps to NA reasoning can consult [DGM10, Section 4]. We briefly sketch here how an NA theory can be translated to a PNA theory. It is simplest to do this if we allow ourselves unknowns with permission sets of the form $\pi{\cdot}\mathbb{A}^<$; by the remark above, this can always be emulated (at some cost in notational convenience).

Consider an axiom $\Delta \vdash r = s$. For each $X \in \mathit{vars}(r,s)$, suppose $\{a \mid a\#X \in \Delta\} = \{a_1,\ldots,a_n\} = A$. Choose some fresh atoms $\{b_1,\ldots,b_n\} = B$. Assign X a permission set $(\mathbb{A}^< \setminus A) \cup B$, where we arrange things such that $A \subseteq \mathbb{A}^<$ and $B \subseteq \mathbb{A}^>$.

Then $\Delta \vdash r = s$ translates to $r = s$.

8. Atoms-abstraction in (P)NA/NEL

The NA in [GM07] included abstraction as primitive; the (P)NA in this paper does not. That atoms-abstraction can be treated as a theory in equality and freshness is a matter of reading and understanding the definitions, such as equation (35) in [GP01] or the reduction rule $(\approx?\textit{-abstraction-}2)$ from Figure 3 in [UPG04].

The observation is implicit in the structure of nominal algebra: see e.g. the $(\mathbf{perm})$ rule of CORE in [GM07] which does not mention abstraction or the two example derivations on page 6, or e.g. Lemma 7.3 part 4 of [GM06a]. A subsection is devoted to the topic in [GM09a, Section 5.1].

NEL underwent a similar development. Abstraction sorts are absent in [CP07] but can be recovered from the more complex non-equivariant operation symbols. The observation that abstraction is a theory was made explicit in [Clo10], which is based on Clouston's PhD, see e.g. [Clo09, Example 4.3.4] and surrounding text.

Having atoms-abstraction is convenient, so we can express in the signature that a term-former binds (like $\lambda([a]r)$). As Clouston and Pitts note "atom-abstraction arities could be added to NEL and probably should be, since making binding information part of a signature rather than part of a theory's axioms is a good idea" [CP07, Section 11]. There are some other good reasons to include atoms-abstraction:

- NA/NEL can express by a simple axiom that a term-former of arity $(\nu,\tau)\tau$ binds, but not that it is *precisely* atoms-abstraction (this is also discussed in [Clo10, Section 8], and see below).
  This seems to matter. For instance, in [Gab09a] which proved an HSPA theorem for nominal algebra, atoms-abstraction emerged from denotational considerations. The proof of the HSPA theorem used a free algebra construction, so to prove HSPA it seems necessary to consider terms with atoms-abstraction, even if the original logic had not included it.
  Kurz and Petrişan revisited this theorem [KP10] using their own methods, and they too encountered some difficulties with atoms-abstraction (see their Section 5).
- Not only abstraction can be axiomatised in NA—so can permutation, substitution, or atom-for-atom renaming. For some applications it might even be useful to drop permutations entirely, but to keep binding; for instance, if we only care about ground terms.
  A host of design decisions are involved here. Yet all the original applications of nominal techniques involved atoms-abstraction, so in the presence of all these choices we should verify in detail—and without any handwaving—that the most common amongst them, works.

This is why atoms-abstraction has been taken as primitive in NA publications.

[16] We also allow atoms-abstraction, non-equivariant constant symbols, and certain classes of infinite permutations, all of which is off-topic for this paper.

Interestingly, PNA enriched with a class of infinite permutations which we call *shift*-permutations, satisfies a stronger HSP theorem [Gab12b]. Permissive-nominal syntax with *shift* has strictly greater expressivity than (P)NA/NEL. Using *shift* has other advantages; this is a topic of current research.

As a distinct issue, Clouston and Pitts's semantic framework of Nominal Lawvere Theories is unable to account directly for atoms-abstraction. Clouston confuses two distinct issues when he presents this in terms of a hypothetical NA/NEL split in [Clo11] "a[n] interesting open question is whether a compelling Lawvere theoretic account can be developed directly for some of the design choices of NA, most notably explicit binding sorts".

NA and NEL have both been considered with and without atoms-abstraction; [GM09a, Subsection 5.1] and [Clo10, Section 6] discuss this and show that expressivity is not lost or gained either way. The issue is not with the language or signature.

The issue here is with the extremely rich structure of nominal sets. Nominal Lawvere Theories are not currently expressive enough to capture the notion 'atoms-abstraction, precisely'. We cannot express a negative assertion that we *only* have binding (that is, a theory can express that an atom is abstracted, but cannot specify that that is *all* that happens).

This 'negative' part of atoms-abstraction cannot be captured by NA/NEL, either, so NA/NEL is not able to detect whether a term-former that claims to be atoms-abstraction, really 'is' atoms-abstraction; this is why expressivity is not lost whether we hard-wire atoms-abstraction into the derivation system or simply write axioms.

As Clouston rightly goes on to remark, if we had a total destructor for atoms-abstraction (instead of the partial destructor given by atoms-concretion) then the framework might be expressive enough to identify 'real' atoms-abstraction.

On this topic, the *nominal renaming sets* of [GH08] are a relevant structure, and it is current research by the author with Dowek to apply these to the family of languages considered in this paper.

9. Other 'nominal' algebras

Moving away from NEL/(P)NA, two other logics with broadly similar goals are under development, and we will briefly sketch them.

Support encoded as multiple sorts. Kurz and Petrişan encode nominal sets in many-sorted first-order algebra [KP10]. For each finite supporting set they take a sort, and they write a collection of axioms describing how permutations interact with sorts/supporting sets, and with term-formers. In this way, it is possible to translate any nominal algebra theory to a(n infinite, but finitely-presentable) first-order theory.

The immediate reason for doing this was to use theorems of first-order model theory to obtain theorems about nominal models, such as the HSP theorem. However, this might also turn out to be a suitable method for designing purpose-built logics for reasoning on nominal sets.

Broadly speaking, the message here is: 'analysis of nominal reasoning using many-sorted first-order syntax and semantics'.

Term equational logic. Fiore and Hur have developed a general framework for generating syntax, equational logic, and sound and complete equational systems [FH08]. The kinds of logics considered in this paper are claimed to be special cases of this framework (though the mathematics is sufficiently complex that this would probably be quite hard to make completely formal). Broadly speaking this seems plausible, with the caveat that if this is so then the framework of Fiore and Hur could probably also be obtained by adding axioms to the base logics here.

The real interest of [FH08], it seems to this author, is the attempt to explicitly parameterise *everything*—syntax, semantics, and proofs—over diagrams in category theory. This is in keeping with Fiore's general style of mathematics, and with a broader current in theoretical computer science, running parallel to the 'sets and functions between them' approach favoured by this author, to present everything as far as possible in abstract, categorical terms. The message here

is: 'parameterise everything as diagrams'. We do not see that there is any real difference in generality, but there is a marked difference in style, and if this different approach can be made to work then it may be possible to apply, in the style of category theory, general diagrammatic arguments to obtain concise proofs of theorems.

10. Conclusion

We have sketched out a design space for equational logics over nominal terms—what a mathematician would call *algebra* [Coh65]. A number of design choices have arisen: semantic freshness or syntactic freshness; atoms-abstraction or no atoms-abstraction; equivariant function-symbols or non-equivariant function-symbols; freshness contexts or permission sets.

These choices are equivalent in expressivity, though PNA has a significantly different treatment of freshness and potentially of α-equivalence.

The goal, when nominal algebra was designed in 2005, was to produce something simple, clear, user-friendly, and compatible with previous work on nominal terms and nominal rewriting [UPG04, FG07]. This author's long-standing belief is that we succeeded: NA and PNA are the simplest possible core logics in their class. Other design decisions are certainly possible, and we do not necessarily promote NA and PNA directly for implementation—but the natural variations add complexity without increasing expressivity, and they do not seem to improve on or greatly change the character of the logic.

We believe that this paper offers the best overview of this family of logics, their clearest and most direct presentation, and the most accurate and best-informed commentary on their design, in existence at the time of writing.[17] We hope the results and discussion here will help readers to understand the design of these logics, and that our detailed discussion will help to inform the design of future generations of logics based on nominal sets. For the interested reader, we collect here a partial selection of the literature: [GP01, GM06a, GM06c, GM07, CP07, FH08, GM09a, GM09b, Clo09, KP10, DGM10, Gab11, Clo11, DG12, Gab12b, Gab12a].

An additional design choice not discussed in this paper is whether to have finite or infinite permutations[18] and whether to allow infinitely-supported constants. This *does* increase expressivity; see [Gab12b, Gab12a].

Permissive-nominal algebra (PNA) was introduced in [GM09b] in 2009 and is sketched in this paper and described in detail in [Gab12b]. This may be the simplest 'nominal algebra' yet. In the permissive-nominal syntax, judgements take the self-evidently purely equational form $r = s$ and the derivation rules are even simpler than those of NA. PNA is the author's currently preferred nominal equational logic, though NA is more familiar and somewhat easier to define (no arbitrary partitions of $\mathbb{A}$ to confuse the reader).

Understanding semantic freshness goes back to [GP01] and its defining equality for freshness (equation 13)

$$a \notin \mathit{supp}(x) \Leftrightarrow \mathsf{И}b.(b\ a)\cdot x = x.$$

$\mathsf{И}$ (meaning 'for all but finitely many') delivers the kind of freshness that syntactic freshness does: something guaranteed sufficiently fresh, but possibly fresher than we absolutely need (cf. Lemma 5.3). This author and Mathijssen designed NA with exactly this in mind.

(P)NA/NEL all have semantics in nominal sets, and because they are all equivalent they also have semantics in Nominal Lawvere Theories. It would be interesting to reconsider Nominal Lawvere Theories in a permissive-nominal context. In particular, we speculate that the ordering on permission sets and *shift*-permutation discussed e.g. in [Gab12b] might be one way to generate a destructor for atoms-abstraction and so extend (permissive) Nominal Lawvere Theories to atoms-abstraction.

[17] . . . for which I am indebted to two anonymous referees, without whose tireless constructive critisism this paper would have been far less rigorous.

[18] Even if finitely representable in some implementation or mathematical foundation, a permutation can still permute infinitely many atoms—think of the operation 'successor' on integers.

Acknowledgements

I am grateful for the detailed input of an editor and of two anonymous referees, and for the support of the Leverhulme trust.

References

[Bou70] Nicolas Bourbaki. *Théorie des ensembles*. Hermann, Paris, 1970.

[CF08a] Chrisophe Calvès and Maribel Fernández. Nominal matching and alpha-equivalence. In *Proceedings of 15th Workshop on Logic, Language and Information in Computation (WoLLIC 2008)*, volume 5110 of *Lecture Notes in Artificial Intelligence*. Springer, June 2008.

[CF08b] Christophe Calvès and Maribel Fernández. A polynomial nominal unification algorithm. *Theoretical Computer Science*, 403:285–306, August 2008.

[Clo09] Ranald Clouston. *Equational logic for names and binding*. PhD thesis, University of Cambridge, UK, 2009.

[Clo10] Ranald Clouston. Binding in nominal equational logic. In *Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2010)*, volume 265 of *Electronic Notes in Theoretical Computer Science*, September 2010.

[Clo11] Ranald Clouston. Nominal Lawvere theories. In *Proceedings of the 18th International Workshop on Logic, Language, and Information (WoLLIC)*, volume 6642 of *Lecture Notes in Computer Science*. Springer, 2011.

[Coh65] P.M. Cohn. *Universal Algebra*. Harper and Row, New York, 1965.

[CP07] Ranald A. Clouston and Andrew M. Pitts. Nominal equational logic. In *Computation, Meaning and Logic: Articles dedicated to Gordon Plotkin*, volume 172 of *Electronic Notes in Theoretical Computer Science*, pages 223–257. Elsevier Science, 2007.

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

[dB72] Nicolaas G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 5(34):381–392, 1972.

[DG10] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic. In *Proceedings of the 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP 2010)*, pages 165–176, 2010.

[DG12] Gilles Dowek and Murdoch J. Gabbay. Permissive Nominal Logic (journal version). *Transactions on Computational Logic*, 2012. In press.

[DGM09] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *Proceedings of the 24th Italian Conference on Computational Logic (CILC'09)*, 2009.

[DGM10] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification: an infinite, co-infinite approach to nominal techniques (journal version). *Logic Journal of the IGPL*, 18(6):769–822, 2010.

[FG07] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting (journal version). *Information and Computation*, 205(6):917–965, June 2007.

[FGM04] Maribel Fernández, Murdoch J. Gabbay, and Ian Mackie. Nominal Rewriting Systems. In *Proceedings of the 6th ACM SIGPLAN symposium on Principles and Practice of Declarative Programming (PPDP 2004)*, pages 108–119. ACM Press, August 2004.

[FH08] Marcelo Fiore and Chung-Kil Hur. Term equational systems and logics. *Electronic Notes in Theoretical Computer Science*, 218:171–192, 2008.

[Gab05] Murdoch J. Gabbay. Axiomatisation of first-order logic (talk). In *Second workshop on Computational Aspects of Nominal sets (CANS'05)*. King's College, London, December 2005.

[Gab09a] Murdoch J. Gabbay. Nominal Algebra and the HSP Theorem. *Journal of Logic and Computation*, 19(2):341–367, April 2009.

[Gab09b] Murdoch J. Gabbay. A study of substitution, using nominal techniques and Fraenkel-Mostowski sets. *Theoretical Computer Science*, 410(12-13):1159–1189, March 2009.

[Gab11] Murdoch J. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[Gab12a] Murdoch J. Gabbay. Finite and infinite support in nominal algebra and logic: nominal completeness theorems for free. *Journal of Symbolic Logic*, September 2012.

[Gab12b] Murdoch J. Gabbay. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*, volume 17. Kluwer, 2012.

[GH08] Murdoch J. Gabbay and Martin Hofmann. Nominal renaming sets. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008)*, pages 158–173. Springer, November 2008.

[GM06a] Murdoch J. Gabbay and Aad Mathijssen. Capture-avoiding Substitution as a Nominal Algebra. In *ICTAC 2006: Theoretical Aspects of Computing*, volume 4281 of *Lecture Notes in Computer Science*, pages 198–212, November 2006.

[GM06b] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. In *18th Nordic Workshop on Programming Theory*, October 2006.

[GM06c] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *Proceedings of the 8th ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP 2006)*, pages 189–200. ACM, July 2006.

[GM07] Murdoch J. Gabbay and Aad Mathijssen. A Formal Calculus for Informal Equality with Binding. In *WoLLIC'07: 14th Workshop on Logic, Language, Information and Computation*, volume 4576 of *Lecture Notes in Computer Science*, pages 162–176. Springer, July 2007.

[GM08] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM09a] Murdoch J. Gabbay and Aad Mathijssen. Nominal universal algebra: equational logic with names and binding. *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 *Proceedings of the 4th International Workshop on Logical Frameworks and Meta-Languages (LFMTP 2009)*, pages 64–73. ACM, August 2009.

[GM10] Murdoch J. Gabbay and Aad Mathijssen. A nominal axiomatisation of the lambda-calculus. *Journal of Logic and Computation*, 20(2):501–531, April 2010.

[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, July 2001.

[HMT85] Leon Henkin, J. Donald Monk, and Alfred Tarski. *Cylindric Algebras*. North Holland, 1971 and 1985. Parts I and II.

[Hod93] Wilfrid Hodges. *Model theory*. Cambridge University Press, 1993.

[KP10] Alexander Kurz and Daniela Petrişan. On universal algebra over nominal sets. *Mathematical Structures in Computer Science*, 20:285–318, 2010.

[LV10] Jordi Levy and Mateu Villaret. An efficient nominal unification algorithm. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA 2010)*, volume 6 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 209–226. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[LV11] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. *Transactions on Computational logic (TOCL)*, 13, 2011.

[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, September 2004.