Universal algebra over lambda-terms and nominal terms: the connection in logic between nominal techniques and higher-order variables

Murdoch J. Gabbay Dominic P. Mulligan

Abstract

This paper develops the correspondence between equality reasoning with axioms using λ-terms syntax, and reasoning using nominal terms syntax. Both syntaxes involve name-abstraction: λ-terms represent functional abstraction; nominal terms represent atoms-abstraction in nominal sets.

It is not evident how to relate the two syntaxes because their intended denotations are so different. We use universal algebra, the logic of equational reasoning, a logical foundation based on an equality judgement form which is spartan but which is sufficiently expressive to encode mathematics in theory and practice.

We investigate how syntax, algebraic theories, and derivability relate across λ-theories (algebra over λ-terms) and nominal algebra theories.

Categories and Subject Descriptors F.4.1 [*Mathematical logic*]

Keywords Universal algebra, lambda-theories, nominal algebra, permissive nominal terms.

1. Introduction

λ-terms are a useful and important syntax in computer science. Nominal terms, from [32], have been more recently developed. What makes both λ-term syntax and nominal term syntax interesting is that they represent name-binding, but they do this in two different ways:

- The basic equality for λ-terms is $\alpha\beta(\eta)$-equivalence. The notion of substitution is capture-avoiding. The standard semantics of name-abstraction is functional abstraction.
- The basic equality for nominal terms is a generalised form of α-equivalence (without β or η). The notion of substitution is capturing. The standard semantics of name-abstraction is non-functional, and uses an atoms-abstraction construction in Fraenkel-Mostowski sets (documented in [19]).

Universal algebra, the theory of equality, is perhaps the simplest non-trivial logics. This paper develops the correspondence between derivability in equational theories over λ-terms (**λ-theories**), and derivability in equational theories over nominal terms (**nominal algebra theories**).

This is the simplest forum within which to address the following question, which has been of interest right from the introduction of nominal techniques in [10, 19]:

> "What connections can be made between reasoning using names and binding based on λ-abstraction, and between reasoning using names and binding based on nominal atoms-abstraction?"

For illustration, we consider some statements about first-order logic and λ-calculus, as written in the informal but rigorous language of mathematical discourse, and translate them to equalities on λ-terms and on permissive nominal terms:

- "If a does not occur in ϕ then $\forall a.\phi$ if and only if ϕ."
- "If a does not occur in ϕ then $\forall b.\exists a.\phi$ if and only if $\exists a.\forall b.\phi$."
- "If a does not occur in t then $\lambda a.(ta)$ is equal to t."
- "$(\lambda b.t)b$ is equal to t always."

Suppose X is a permissive nominal unknown with permissions set $p(X) = comb$, and suppose $b \in comb$ and $a \notin comb$ (so intuitively, X may be instantiated to a term mentioning b free but not mentioning a free; formal definitions are in the body of the paper). Then translations are as in the following small table:

λ-terms	nominal terms
$\forall \lambda a.c = c$	$\forall [a]X = X$
$\exists \lambda a.\forall \lambda b.(cb) = \forall \lambda b.\exists \lambda a.(cb)$	$\exists [a]\forall [b]X = \forall [b]\exists [a]X$
$\mathsf{lam}\lambda a.(ca) = c$	$\mathsf{lam}[a](Xa) = X$
$(\mathsf{lam}\lambda b.c)b = c$	$(\mathsf{lam}[b]X)b = X$

Here, quantifiers feature as term-formers, and their binding behaviour is handled by (λ/nominal)-abstraction; this is standard. In the case of λ-terms, the possibility of capture is implemented using application. In the case of nominal terms, capture-avoidance is enforced by using names that are not in the permissions set $comb$.

Note here that lam is a term-former intended to represent λ-abstraction — whose behaviour, in the left-hand column but not in the right-hand column, is axiomatised using a λ-abstraction of a λ-terms syntax.

There are no types above, nor in the syntax we develop below. We believe that types could easily be added to the proofs which follow, and then it would be possible for example to make distinctions between variables ranging over a type of truth-values and variables ranging over a type of individuals. To the level of detail that interests us here, an untyped syntax is sufficient.

Some details of our translation between λ-terms syntax and nominal terms syntax are subtle; the translations of substitutions

Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.
LFMTP '09, August 2, 2009, Montreal, Canada.

and theories in particular are delicately assembled. Full details are in the body of the paper.

However, it is not hard to give an overview of the technical ideas involved, and they resonate with other work. We account for the capturing substitution in nominal terms by translating to 'raised' λ-terms, as in previous work [6, 7, 23]. Nominal atoms-abstraction intuitively satisfies α-equivalence (not β or η), so the translation identifies λ-theories with nominal algebra theories containing ULAME, a theory which has been identified in previous work as axiomatising $\alpha\beta\eta$-equivalence [16, 18] (see also Definition 14).

The translation works for 'sufficiently large' finite-variable fragments of both languages. As in [7] we do not present a single fixed translation of terms and derivations and we suspect (but do not prove) that such a translation does not exist. Our results only work for $\alpha\beta\eta$-equivalence: $\alpha\beta$ seems to be harder. Details follow in the body of the paper.

We will now briefly motivate algebra as a canonical 'simple but useful' logic, and discuss issues in the design of algebraic logics.

Algebraic logic. Algebraic logic is the theory of derivable equality between terms. For example, 'universal algebra' [4] is the theory of equality between first-order syntax (syntax made of *variables* and *term-formers*, also called *function symbols*). An algebraic theory is just a collection of equalities which are asserted as axioms, and any model of an algebraic theory should interpret derivably equal terms as denotationally identical elements.

The judgement-form of algebra is $r = s$. This might seem limited; for example there is no implication, so we cannot say 'if' or 'but'. Yet surprisingly complex theories can be expressed using just equalities, and then the limitations become an asset: if we know that $r = s$ then we can interchange r and s — with no 'if's and 'but's. This is particularly useful for compositional reasoning, and that may be one reason that equality reasoning has been chosen as the foundation of some large-scale implemented reasoning systems like [20]. Also, logics of equality enjoy powerful mathematical properties, like the HSP theorem [4, 11], which can be exploited to prove non-trivial properties of algebraically-specified logic and computation [1, 25, 29].

So universal algebra is interesting and important; in this paper, we use equality as a logical judgement-form with which to explore the implications of reasoning on different syntaxes with names and binding.

Choosing a syntax for algebra: first-order, higher-order, and nominal. When we design an algebraic logic we cannot choose our logical connectives — because there are none — but we have freedom in what syntax we use for r and s. So what syntax should we use?

Equality over just first-order syntax is a suitable framework for some exceedingly complex mathematics; consider for example combinators and their theory. It is also possible to study systems with binding using the so-called *cylindric* method [22]. This fixes a term-former for each binder; for example the spirit of *cylindric algebra* is that we axiomatise first-order logic in univeral algebra, and this has one term-former for $\forall x$, another for $\forall y$, and so on; likewise Salibra's λ-abstraction algebras [25, 30] axiomatise λ-calculus in universal algebra, and they have one term-former for λx, another for λy, and so on (so cylindric algebra and λ-abstraction algebras use infinitely many term-formers, and infinitely many axioms).

However, it is still important to consider algebra over syntax with primitive support for binding. This is motivated by the need for the mathematical study of informal practice, and the creation of computing systems which match informal practice *and* are backed by rigorous formal methods.

One syntax extending first-order syntax with binding is λ-terms syntax quotiented by $\alpha\beta$- or $\alpha\beta\eta$-equivalence. This is relevant to practice as well as theory: μCRL2 is an algebraic reasoning framework [20] which includes functions for usability and natural expressiveness [21]; in his book [2] Peter Andrews presents a logic $\mathcal{Q}_0$ for mathematics as an equational theory on (typed) λ-terms; *λ-theories* are equational theories on λ-calculus syntax including $\alpha\beta$-equivalence, studied by mathematicians, e.g. [24].

Nominal terms [32] provide primitive support for binding, via a generalisation of α-binding. Nominal algebra [17] is the logic of equality on nominal terms, introduced in [12, 13, 26]. It has been tested as a logical framework, at least in principle, by using it to axiomatise the λ-calculus and first-order logic [15, 18].

λ-terms' natural denotation uses functions. In contrast, nominal terms admit a denotation in a cumulative sets hierarchy in which name-abstraction is interpreted using a non-functional abstraction operation described in [19] with properties much like those of an α-equivalence class (we do not consider denotations in this paper).

Translations between nominal unification [32] and higher-order patterns [28] have been considered; by Cheney [5], Levy and Villaret [23], and most recently by Dowek and the authors [6, 7]. This recent work uses a *permissive* variant of nominal terms, which offers advantages documented in [6, 7] and in this paper; to a first approximation we can ignore the difference, but this paper is self-contained and includes all relevant details of the permissive nominal syntax.

A difference between [23] and [7] is that [23] translates unifiability whereas [7] translates unification problems and their solutions. Now a substitution solving a unification problem is in a sense a 'derivation' or 'proof-object' for its unifiability.

In this paper we use the same basic ideas as in [7], but we extend them and take on the more complex task of translating derivations of equality in the presence of axioms (i.e. we translate derivations of equality modulo an arbitrary theory). This adds some technical complexity which the permissive nominal framework handles very well, but beyond that, this paper gives the details of a translation between logical reasoning over a functional syntax expressive enough to be a framework for mathematics, and a similarly expressive one over a nominal syntax.

2. Algebra-with-binding: nominal algebra and λ-theories

2.1 Permissive nominal algebra

We define permissive nominal terms (Definition 7). This features a *two-level syntax* inherited from [32], and a capturing substitution (Definition 10). We define an equational theory on permissive nominal terms (Definition 13); the notions of equational theory and derivability are are based on those in [12, 14, 17], but here we use a permissive nominal syntax. Proving properties of the connection between nominal algebra and the permissive variant in this paper, is future work; it will be similar to the relation between nominal unification and permissive nominal unification [7].

We define an equational theory ULAME (Definition 14); it is shown in [18] that we can think of this as '$\beta\eta$ for nominal atoms-abstraction'. We will need this theory because our translation identifies the (weak) nominal atoms-abstraction with the (relatively strong) λ-terms functional abstraction.

Definition 1. Fix a countably infinite set $\mathbb{A}$ of **atoms**. $a, b, c, \ldots$ will range over *distinct* atoms (we call this the **permutative convention**). Fix a set of **term-formers**. $\mathsf{f}, \mathsf{g}, \mathsf{h}$ will range over distinct term-formers.

Definition 2. If f is a function from atoms to atoms define $nontriv(f) = \{a \mid f(a) \neq a\}$.[1]

Definition 3. A **permutation** π is a bijection on atoms such that $nontriv(\pi)$ is finite. π and π' will range over permutations (not necessarily distinct).

Write $\pi \circ \pi'$ for the **composition** of π and π' (so $(\pi \circ \pi')(a) = \pi(\pi'(a))$). Write id for the **identity** permutation (so $id(a) = a$ always). Write $(a\ b)$ for the **swapping** permutation that maps a to b and b to a and all other c to themselves.

Definition 4. If $S \subseteq \mathbb{A}$, define the **pointwise** action by:

$$\pi \cdot S = \{\pi(a) \mid a \in S\}$$

Definition 5. Call $S \subseteq \mathbb{A}$ co-infinite when $\mathbb{A} \setminus S$ is infinite. Fix an infinite, co-infinite set $comb \subseteq \mathbb{A}$.

A **permission set** has the form $\pi \cdot (comb \setminus A)$ for a finite set $A \subseteq \mathbb{A}$. S, S', T will range over permission sets.[2]

Definition 6. For each permission set S fix a disjoint countably infinite set of **unknowns** of sort S. X, Y, Z, will range over distinct unknowns. We write $p(X)$ for the permission set of X.

Definition 7. Define **(permissive nominal) terms** by:

$$r, s, t, \ldots ::= a \mid \pi \cdot X \mid \mathsf{f} \mid [a]r \mid r'r$$

We write $\equiv$ for syntactic identity; $r \equiv s$ when r and s denote identical terms.

Note that X (the unknown) is not a term, however $\pi \cdot X$ is a term and in particular $id \cdot X$ is a term, which we may write as X.

The intended interpretation of permissive nominal terms follows the interpretation of nominal terms [32]: atoms represent *variables*; term-formers *functions*; unknowns *meta-variables*; abstraction $[a]r$ *binding*; and $\pi \cdot X$ a meta-variable with a suspended substitution, like '$t[y/x]$'. The application $r'r$ in this syntax is *a priori* just a convenient way to attach terms together; we could just as well have terms like $\mathsf{f}(r_1, \ldots, r_n)$. For our purposes in this paper, it is more convenient to take as primitive a binary operator.

Definition 8. Define a **permutation action** by:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot \mathsf{f} \equiv \mathsf{f} \qquad \pi \cdot (r'r) \equiv (\pi \cdot r')(\pi \cdot r)$$
$$\pi \cdot [a]r \equiv [\pi(a)](\pi \cdot r) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X$$

Definition 9. Define **free atoms** $fa(r)$ by:

$$fa(a) = \{a\} \qquad fa(\mathsf{f}) = \varnothing \qquad fa(r'r) = fa(r') \cup fa(r)$$
$$fa([a]r) = fa(r) \setminus \{a\} \qquad fa(\pi \cdot X) = \pi \cdot p(X)$$

Note that $fa(\pi \cdot X) = \pi \cdot p(X)$. An intuition for $fa(r)$ is 'the free atoms we can have after instantiation'. Since r may contain unknowns X, this need not be finite, though for each complete instantiation of r the free atoms will be finite.

Definition 10. A **substitution** θ is a function from unknowns to terms such that $fa(\theta(X)) \subseteq p(X)$ always (so $p(X)$ describes the 'permission' we have to instantiate X, namely to terms with free atoms in S).[3]

θ will range over substitutions.

Write id for the **identity** substitution mapping X to $id \cdot X$ always. It will be clear whether id means the identity substitution or permutation.

Suppose $fa(t) \subseteq p(X)$. Write $[X{:=}t]$ for the substitution such that $[X{:=}t](X) \equiv t$ and $[X{:=}t](Y) \equiv id \cdot Y$ for all other Y.

Definition 11. Define a **substitution action** on terms by:

$$a\theta \equiv a \qquad \mathsf{f}\theta \equiv \mathsf{f} \qquad (r'r)\theta \equiv (r'\theta)(r\theta)$$
$$([a]r)\theta \equiv [a](r\theta) \qquad (\pi \cdot X)\theta \equiv \pi \cdot \theta(X)$$

Consistent with [32] substitution for unknowns is capturing for abstraction by atoms. Note that $X\theta$ means 'θ acting on $id \cdot X$' and $\theta(X)$ means 'the function θ at X'.

We now set about defining permissive nominal algebra.

Definition 12. An **axiom** is a pair $r = s$. A **(permissive nominal algebra) theory** is a set of axioms. T will range over theories.

Definition 13. Define **derivable equality** $=$ by:

$$\frac{r = s \quad s = t}{r = t}\ (\mathbf{Ntran}) \qquad \frac{}{r = r}\ (\mathbf{Nrefl}) \qquad \frac{r = s}{s = r}\ (\mathbf{Nsym})$$

$$\frac{r = s}{[a]r = [a]s}\ (\mathbf{Nabs}) \qquad \frac{r' = s' \quad r = s}{r'r = s's}\ (\mathbf{Napp})$$

$$\frac{(a, b \notin fa(r))}{r = (a\ b) \cdot r}\ (\mathbf{N}\alpha) \qquad \frac{}{\pi \cdot (r\theta) = \pi \cdot (s\theta)}\ (\mathbf{Nax}^{\pi,\theta}_{\mathbf{r=s}})$$

If T is a theory, write $\mathsf{T} \vdash r = s$ when $r = s$ is derivable, using only instances $(\mathbf{Nax}^{\pi,\sigma}_{\mathbf{r'=s'}})$ where $(r' = s') \in \mathsf{T}$.

Π will range over derivations.

Definition 14. Let $a, b \in comb$. Let X, Z, and Z' have permission set $comb$. Let Y' have permission set $comb \setminus \{b\}$. Let X' have permission set $comb \setminus \{a\}$. Define a theory ULAME by:

$([a]a)X = X$	$(\mathbf{N}\beta\mathbf{var})$
$([a]X')X = X'$	$(\mathbf{N}\beta\#)$
$([a]Z)a = Z$	$(\mathbf{N}\beta\mathbf{id})$
$([a]([b]Z))Y' = [b](([a]Z)Y')$	$(\mathbf{N}\beta\mathbf{abs})$
$([a](Z'Z))X = (([a]Z')X)(([a]Z)X)$	$(\mathbf{N}\beta\mathbf{app})$
$[a](X'a) = X'$	$(\mathbf{N}\eta)$

Remark 15. ULAME is a set with six elements. It is not an infinite set determined by axiom-schemes.

The choice of atoms and unknowns in the axioms of ULAME is not important because $(\mathbf{Nax})$ has facilities to rename atoms permutatively using a permutation π, and to instantiate unknowns using a substitution θ. The choice of atoms and unknowns in the axioms of ULAME can be important for the technical details of the proofs; see for example the case of $(\mathbf{L}\beta\#)$ in Theorem 51.

ULAME originates in [16, 18], where the theory is proved sound and complete for λ-terms quotiented by $\alpha\beta\eta$-equivalence (with no other equalities).

Example 16. Suppose $p(X){=}p(Y){=}comb$ and $a, b{\in}comb$. We sketch two ULAME derivations in calculational style; the full

[1] We call this the *support* of π in [19]. There, 'support' has a specific technical meaning in the universe of Fraenkel-Mostowski sets. Here, we prefer $nontriv$ (for 'nontrivial') because the terminology immediately suggests its definition.

[2] Two notes about previous work:
Nominal sets [19] disallow $comb$ because it lacks *finite support*; here, we are working at the meta-level, and we can talk about any subset we like.
In [6, 7] we allow slightly more permissions sets, of the form $(comb \cup B) \setminus A$ for finite B. What we have defined here, will be enough for our needs.

[3] '$fa(\theta(X)) \subseteq p(X)$' looks absent in nominal terms theory ([32, Definition 2.13], [9, Definition 4]), yet it is there: see the conditions '$\nabla' \vdash \theta(\nabla)$' in Lemma 2.14, and '$\nabla \vdash a\#\theta(t)$' in Definition 3.1 of [32].

derivation trees can easily be reconstructed:

$$\begin{array}{ll}([b]([a]X))Y = ([b]([a']([a'\ a)\cdot X))Y & (\mathbf{N}\alpha)\ a, a' \# fa([a]X) \\ = [a'](([b]((a'\ a)\cdot X))Y) & (\mathbf{N}\beta\mathbf{abs}) \end{array}$$

$$\begin{array}{ll}([b]([a]b))a = ([b]([a']b))a & (\mathbf{N}\alpha)\ a, a' \# fa([a]b) \\ = [a'](([b]b)a) & (\mathbf{N}\beta\mathbf{abs}) \\ = [a']a & (\mathbf{N}\beta\mathbf{var}) \end{array}$$

Here, we choose a' fresh, so $a' \notin comb$ (and by our permutative convention a' is not equal to a or b).

Definition 17. Define $fV(r)$ by:

$$fV(a) = \varnothing \qquad fV(\mathsf{f}(r_1, \ldots, r_n)) = fV(r_1) \cup \cdots \cup fV(r_n)$$
$$fV([a]r) = fV(r) \qquad fV(\pi \cdot X) = \{X\}$$

Lemma 18 is a key *difference* from nominal terms. Later, when we write 'choose a fresh', we are using Lemma 18:

Lemma 18. *$\mathbb{A} \setminus fa(r)$ is always infinite.*

Proof. By induction on r. We consider one case: $fa(\pi \cdot X) = \pi \cdot p(X)$ and by assumption $\mathbb{A} \setminus (\pi \cdot p(X))$ is infinite. □

The following lemmas are proved by inductions just as in [32]:

Lemma 19. *$id \cdot r \equiv r$, and $\pi' \cdot (\pi \cdot r) \equiv (\pi' \circ \pi) \cdot r$.*

Lemma 20. *$\pi \cdot fa(r) = fa(\pi \cdot r)$.*

Lemma 21. *$\pi \cdot (r\theta) \equiv (\pi \cdot r)\theta$.*

2.2 λ-theories

Just as we did for nominal terms, we define λ-terms syntax and their (standard) capture-*avoiding* substitution action (Definitions 22 and 26). We define a notion of derivable equality (Definition 28) up to $\alpha\beta\eta$-equality, plus equational axioms; this is similar to Andrews's equational axiomatisation $\mathcal{Q}_0$ [2] and to the axioms of Salibra's Lambda-Abstraction Algebras [30].

Note that we *need* η-equivalence for our later proofs to work. The use of η-equivalence is consistent with other work, e.g. on higher-order unification [8]. η-equivalence does sometimes make theory easier; this paper is a case in point. We do not know if our constructions can be made to work without η, perhaps in a more complicated form.[4]

Definition 22. Define **λ-terms** by:

$$g, h, \ldots ::= a \mid X \mid \mathsf{f} \mid \lambda a.g \mid g'g$$

Here f ranges over term-formers, a ranges over atoms, and X ranges over unknowns. g, h, k will range over λ-terms.

Definition 23. Define a **permutation action** by:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot X \equiv X \qquad \pi \cdot \mathsf{f} \equiv \mathsf{f}$$
$$\pi \cdot (\lambda a.g) \equiv \lambda \pi(a).(\pi \cdot g) \qquad \pi \cdot (g'g) \equiv (\pi \cdot g')(\pi \cdot g)$$

Definition 24. Define **free atoms** by:

$$fa(a) = \{a\} \qquad fa(X) = \varnothing \qquad fa(\mathsf{f}) = \varnothing$$
$$fa(\lambda a.g) = fa(g) \setminus \{a\} \qquad fa(g'g) = fa(g') \cup fa(g)$$

Definition 25. Call a function σ from unknowns to λ-terms a (λ-calculus) **substitution**. σ will range over substitutions.

Write $[X{:=}h]$ for the substitution which maps X to h and maps all other Y to Y.

[4] To be specific: we use η-equivalence in Lemma 58. We need something like this result for Theorems 62 and 65. If we drop η then we should modify the translations $[\![\text{-}]\!]^D$ and $(\!|\text{-}|\!)^D$ to be 'more intelligent', i.e. more complicated, to compensate so that the η-equivalence of Lemma 58 becomes syntactic identity or perhaps an $\alpha\beta$-equivalence.

Definition 26. Define the **capture-avoiding substitution** action $g\sigma$ on λ-terms by:

$$a\sigma \equiv a \qquad X\sigma \equiv \sigma(X) \qquad \mathsf{f}\sigma \equiv \mathsf{f} \qquad (g'g)\sigma \equiv (g'\sigma)(g\sigma)$$
$$(\lambda a.g)\sigma \equiv \lambda a.(g\sigma) \qquad (a \notin \bigcup\{fa(\sigma(X)) \mid X \in fV(g)\})$$
$$(\lambda a.g)\sigma \equiv \lambda b.(((b\ a)\cdot g)\sigma) \qquad (a \in \bigcup\{fa(\sigma(X)) \mid X \in fV(g)\})$$

Definition 27. An **axiom** is a pair $g = h$. A **(λ-)theory** is a set of axioms. L will range over theories.

Definition 28. Define **derivable equality** by:

$$\frac{g = g' \quad g' = g''}{g = g''}\ (\mathbf{Ltran}) \qquad \frac{}{g = g}\ (\mathbf{Lrefl}) \qquad \frac{g = h}{h = g}\ (\mathbf{Lsym})$$

$$\frac{}{(\lambda a.a)g = g}\ (\mathbf{L}\beta\mathbf{var}) \qquad \frac{(a \notin fa(h))}{(\lambda a.h)g = h}\ (\mathbf{L}\beta\#)$$

$$\frac{}{(\lambda a.h)a = h}\ (\mathbf{L}\beta\mathbf{id}) \qquad \frac{(b \notin fa(g))}{(\lambda a.(\lambda b.h))g = \lambda b.((\lambda a.h)g)}\ (\mathbf{L}\beta\lambda)$$

$$\frac{}{(\lambda a.(h'h))g = ((\lambda a.h')g)((\lambda a.h)g)}\ (\mathbf{L}\beta\mathbf{app}) \qquad \frac{(a \notin fa(g))}{\lambda a.(ga) = g}\ (\mathbf{L}\eta)$$

$$\frac{g = h}{\lambda a.g = \lambda a.h}\ (\mathbf{L}\lambda) \qquad \frac{g = g' \quad h = h'}{gh = g'h'}\ (\mathbf{Lapp})$$

$$\frac{(a, b \notin fa(g))}{g = (a\ b)\cdot g}\ (\mathbf{L}\alpha) \qquad \frac{}{g\sigma = h\sigma}\ (\mathbf{Lax}^{\sigma}_{\mathbf{g=h}})$$

Write $\mathsf{L} \vdash g = h$ when $g = h$ is derivable using only instances $(\mathbf{Lax}^{\sigma}_{\mathbf{g'=h'}})$ for $(g' = h') \in \mathsf{L}$. Also:

- Write $g =_{\alpha\beta\eta} h$ when $\varnothing \vdash g = h$.
- Write $g =_{\alpha\beta} h$ when $\varnothing \vdash g = h$ with a derivation that does not use $(\mathbf{L}\eta)$.
- Write $g =_{\eta} h$ when $\varnothing \vdash g = h$ with a derivation that does not use $(\mathbf{L}\alpha)$, $(\mathbf{L}\beta\mathbf{var})$, $(\mathbf{L}\beta\#)$, $(\mathbf{L}\beta\mathbf{id})$, $(\mathbf{L}\beta\lambda)$, or $(\mathbf{L}\beta\mathbf{app})$.
- Write $g =_{\alpha} h$ when $\varnothing \vdash g = h$ with a derivation that does not use $(\mathbf{L}\eta)$, $(\mathbf{L}\beta\mathbf{var})$, $(\mathbf{L}\beta\#)$, $(\mathbf{L}\beta\mathbf{id})$, $(\mathbf{L}\beta\lambda)$, or $(\mathbf{L}\beta\mathbf{app})$.

Λ will range over derivations.

Remark 29. Some comments on the provenance of these rules:

- $(\mathbf{Lrefl})$, $(\mathbf{Lapp})$, $(\mathbf{L}\lambda)$, and $(\mathbf{Ltran})$, have to do with equality and are as one would expect.
- $(\mathbf{Lax}^{\sigma}_{\mathbf{g=h}})$ is the axiom rule. The substitution σ allows us to instantiate unknowns in axioms (atoms can be substituted for using λ-abstraction and β-reduction).
- Other axioms are adapted from Andrews's logic $\mathcal{Q}_0$ [2], which includes equality axioms for β-conversion (though they also look a lot like Salibra's axioms of Lambda-Abstraction Algebras [25, Definition 20], and probably like other similar axiomatisations too).[5]
  $(\mathbf{L}\beta\mathbf{var})$, $(\mathbf{L}\beta\mathbf{id})$, $(\mathbf{L}\beta\lambda)$, and $(\mathbf{L}\beta\mathbf{app})$ are exactly Andrews's axioms $(\mathbf{4_2})$, $(\mathbf{4_4})$, and $(\mathbf{4_3})$. In the presence of α-conversion, rules $(\mathbf{4_3})$ and $(\mathbf{4_4})$ let us 'push a β-reduct into a term' and it is a fact that they give the power of $(\mathbf{L}\beta\#)$.
  $(\mathbf{L}\eta)$ is η-equivalence. Andrews's system does not include η — if it had, it would probably have looked like $(\mathbf{L}\eta)$.
  $(\mathbf{L}\alpha)$ is α-equivalence. Andrews's system includes an extensionality axiom expressed using an axiomatised $\forall$ quantifier ($(\mathbf{3}^{\alpha\beta})$ in [31, page 20]). It is possible to derive every instance of $(\mathbf{L}\alpha)$ in $\mathcal{Q}_0$. Since we do not axiomatise $\forall$, we must include $(\mathbf{L}\alpha)$ (compare also with [25, Definition 20, rule (α)]).

[5] We first obtained the axioms from Tan's thesis [31]. We are grateful for his clear and accessible presentation of Andrews's work.

3. Translating nominal terms into the λ-calculus

We are now ready to translate nominal terms to λ-terms (Definition 35). The translation follows [7] and is indexed by a vector of atoms (Definition 30).

What does this vector mean? A difference between nominal terms and λ-terms is that the former permits capturing substitution and the latter does not. However, as is well-known, the *effect* of a capturing substitution can be obtained in λ-terms by 'raising'. For example, the effect of the nominal term $[a]X$ can be obtained by $\lambda a.(Xa)$, and the effect of the capturing substitution $[X{:=}a]$ on the nominal term $[a]X$ can be obtained by the capture-avoiding substitution $[X{:=}\lambda a.a]$. Here, X is 'raised' over a.

Intuitively, the vector of the translation maps nominal terms to 'raised' λ-terms; we raise over the indexing vector, thus deciding once and for all, for each vector chosen, for which atoms we permit capture. There is no canonical vector to raise over, but if we are given a derivation in nominal algebra then we can raise over the atoms in that derivation and, with a little further work, obtain a valid derivation in λ-calculus. This is Theorem 43.

The translation maps nominal terms atoms-abstraction $[a]r$ to λ-terms functional abstraction $\lambda a.g$. Therefore, we also verify that the axioms in ULAME translate to valid $\alpha\beta\eta$-equivalences between λ-terms (Theorem 45).

Because theories may be infinite, we do not collect all the atoms mentioned in a theory when we decide on an index to translate it. Instead, we calculate the appropriate vector on an axiom-by-axiom basis ($[\![\mathsf{T}]\!]$, Definition 35). One unfortunate side-effect of this is that when we try to use the translation of an axiom in the translation of a derivation, the vectors may not match and we may need to 'reindex'. This is achieved by indexing the translation of substitutions $[\![\theta]\!]^E_D$ in Definition 35 over two vectors of atoms. The same thing happened in [7] but for apparently different technical reasons, and there may be some mathematics here to be brought out proving the inevitability of this, in future work.

3.1 The translation $[\![\text{-}]\!]^D$, and its compositionality

We define the translation in Definition 35, and we prove a compositionality result (Theorem 40) for substitution relative to the translation; as discussed above, it is convenient to build into this result the possibility to 'reindex' from a vector D to a vector E (this did not happen in [7]). Note that compositionality holds only if there are 'enough atoms', where 'enough' varies depending on the complexity of the term and substitution we are considering.

Definition 30. Call a finite list of distinct atoms a **vector**. D will range over vectors. Write $[d_1,\ldots,d_n]$ for the vector containing d_1, $\ldots$, d_n in that order.

Definition 31. We use the following notation: Suppose $A \subseteq \mathbb{A}$.

- Write $D \cap A$ for the vector of atoms in D that occur in A, in order; thus $[a_1,a_2,a_3] \cap \{a_1,a_3,a_5\} = [a_1,a_3]$.
- Write $D \subseteq A$ when every atom in D is in A.
- Write $A \subseteq D$ when every atom in A is in D.
- Write $D \cup A$ for the *set* of atoms appearing in D or A.
- If D and D' are vectors, write $D \subseteq D'$ when every atom in D is contained in D' (not necessarily in the same order).
- If π is a permutation and $D = [d_1,\ldots,d_n]$, write $\pi \cdot D$ for the vector $[\pi(d_1),\ldots,\pi(d_n)]$ (the pointwise action).

Definition 32. Suppose $D = [d_1,\ldots,d_n]$. Write $\lambda D.g$ as shorthand for $\lambda d_1.\ldots.\lambda d_n.g$ (a list of λ-abstractions). Write $g\,D$ as shorthand for $g\,d_1\ldots d_n$ (a list of applications).

Definition 33. Write $atoms(D)$ for the set of atoms appearing in D. Similarly write $atoms(r)$ for the set of atoms appearing in r (for example, $atoms((a\ b)\cdot X) = atoms([a][b]X) = \{a,b\}$); an inductive definition will be in a longer version of this paper. We will extend this notation to other tree-like structures as convenient, including to derivations (as in $atoms(\Pi)$) and to theories (as in $atoms(\mathsf{ULAME})$).

Definition 34. For each finite set of atoms S make a fixed but arbitrary choice of order on S.

Definition 35. Translate a nominal term r to a λ-term $[\![r]\!]^D$ by:

$$[\![a]\!]^D \equiv a \qquad [\![\pi\cdot X]\!]^D \equiv X(\pi\cdot(D\cap p(X)))$$
$$[\![[a]r]\!]^D \equiv \lambda a.[\![r]\!]^D \qquad [\![r'r]\!]^D \equiv [\![r']\!]^D[\![r]\!]^D \qquad [\![\mathsf{f}]\!]^D \equiv \mathsf{f}$$

Extend the translation to substitutions, equalities, and theories by:

$$[\![\theta]\!]^E_D(X) \equiv \lambda(D\cap p(X)).[\![\theta(X)]\!]^E$$
$$[\![r=s]\!]^D \equiv [\![r]\!]^D = [\![s]\!]^D$$
$$[\![\mathsf{T}]\!] = \{[\![r=s]\!]^D \mid r=s\in\mathsf{T}\}$$

In the translation of T, D is $atoms(r)\cup atoms(s)$ in the order chosen in Definition 34.

Lemma 36. *$id\cdot g \equiv g$, and $\pi\cdot(\pi'\cdot g)\equiv(\pi\circ\pi')\cdot g$*

Lemma 37. *$(\lambda a.g)a =_{\alpha\beta} g$.*
$b\notin fa(g)$ implies $(\lambda a.g)b =_{\alpha\beta} (a\ b)\cdot g$.
If $nontriv(\pi)\cap fa(g)\subseteq D$ then $(\lambda D.g)\pi\cdot D =_{\alpha\beta} \pi\cdot g$.

Proof. The first two parts can be proved by inductions on a measure of the size of g.

For the third part, we work by induction on the length of D. If D has one element we use the first part. Suppose D has more than one element. Write $\pi' = (\pi(a)\ a)\circ\pi$, $head(D)=a$, and $tail(D)=D'$ ($head(D)$ is the first element in D and $tail(D)$ is the other elements). There are two cases, depending on whether $\pi(a)=a$ or $\pi(a)\neq a$. We consider only the case $\pi(a)\neq a$; the other is similar.

Note the following facts: $\pi'(a') = \pi(a')$ for every a' in D', $\pi'(a)=a$, $nontriv(\pi')\cap fa(g)\subseteq D'$, and $(\pi(a)\ a)\circ\pi' = \pi$. We reason as follows:

$$\begin{array}{lll}(\lambda D.g)(\pi\cdot D) \equiv (\lambda a.\lambda D'.g)\pi(a)(\pi'\cdot D') & & \text{Facts}\\ =_{\alpha\beta} (a\ \pi(a))\cdot((\lambda D'.g)\pi'\cdot D') & & \text{First part, } \pi(a)\in D'\\ =_{\alpha\beta} (a\ \pi(a))\cdot(\pi'\cdot g) & & \text{Ind. hyp.}\\ \equiv \pi\cdot g & & \text{Lemma 36}\end{array}$$ □

Lemma 38. $[\![\pi\cdot r]\!]^D \equiv \pi\cdot[\![r]\!]^D$

Proof. By induction on r. □

Lemma 39. $fa([\![r]\!]^D)\subseteq fa(r)$.

Proof. By induction on r. We consider one case.

- The case $\pi\cdot X$. We reason as follows, where $[d_1,\ldots,d_n] = D\cap p(X)$:

$$\begin{array}{lll}fa([\![\pi\cdot X]\!]^D) = fa(X\pi(d_1)\ldots\pi(d_n)) & & \text{Definition 35}\\ = fa(\pi(d_1))\cup\ldots\cup fa(\pi(d_n)) & & \text{Definition 24}\\ = \pi\cdot(fa(d_1)\cup\ldots\cup fa(d_n)) & & \text{Fact}\\ \subseteq \pi\cdot fa(X) & & \text{Definition 9}\\ = fa(\pi\cdot X) & & \text{Definition 9}\end{array}$$ □

Theorem 40. *If $atoms(r)\subseteq D$ then $[\![r\theta]\!]^E =_{\alpha\beta} [\![r]\!]^D[\![\theta]\!]^E_D$.*

Proof. By induction on r.

- The case $\pi \cdot X$. $[\![\theta]\!]^E_D(X) = \lambda(D \cap p(X)).[\![\theta(X)]\!]^E$ by Definition 35. Then:

$$\begin{array}{rll}
[\![(\pi\cdot X)\theta]\!]^E \equiv [\![\pi \cdot \theta(X)]\!]^E & & \text{Definition 10}\\
\equiv \pi \cdot [\![\theta(X)]\!]^E & & \text{Lemma 38}\\
=_{\alpha\beta} (\lambda(D\cap p(X)).[\![\theta(X)]\!]^E)(\pi\cdot(D\cap p(X))) & & \text{Lemma 37}\\
\equiv (X(\pi \cdot (D \cap p(X))))[\![\theta]\!]^E_D & & \text{Definition 35}\\
\equiv [\![\pi \cdot X]\!]^D[\![\theta]\!]^E_D & & \text{Definition 35}
\end{array}$$

  The use of Lemma 37 above is valid, since by assumption $nontriv(\pi) \subseteq D$, and so $nontriv(\pi) \cap fa(\theta(X)) \subseteq D$. The result follows.
- The case $[a]r$. Choose b fresh, so $b \notin fa([\![r\theta]\!]^D)$, $b \notin \bigcup_{X\in fV([\![r]\!]^D)} fa([\![\theta]\!]^E_D(X))$, and $b \notin fa([\![r]\!])^D$. Then:

$$\begin{array}{rll}
[\![([a]r)\theta]\!]^E \equiv [\![[a](r\theta)]\!]^E & & \text{Definition 10}\\
\equiv \lambda a.([\![r\theta]\!]^E) & & \text{Definition 35}\\
=_{\alpha} \lambda b.(b\ a)\cdot[\![r\theta]\!]^E & & \text{Lemma 39, } b \text{ fresh}\\
=_{\alpha\beta} \lambda b.(b\ a)\cdot([\![r]\!]^D[\![\theta]\!]^E_D) & & \text{Inductive hypothesis}\\
\equiv (\lambda b.(b\ a)\cdot[\![r]\!]^D)[\![\theta]\!]^E_D & & b \text{ fresh}\\
=_{\alpha} (\lambda a.[\![r]\!]^D)[\![\theta]\!]^E_D & & \text{Lemma 39, } b \text{ fresh}\\
\equiv ([\![[a]r]\!]^D)[\![\theta]\!]^E_D & & \text{Definition 35}
\end{array}$$

  The result follows. □

Example 41. Suppose $a, a' \in comb$ and $b \notin comb$. Consider $r \equiv [b](b\ a)\cdot X$ where $p(X) = comb$. Note that $atoms(r) = \{a, b\}$. Take $D = [a, b]$ and $E = [a, a', b]$. It is a fact that $[\![r]\!]^D \equiv \lambda b.(Xb)$.

Take $\theta = id$. Then $[\![\theta]\!]^E_D(X) \equiv \lambda a.(Xaa')$, and

$$\begin{array}{rl}
[\![r]\!]^D[\![\theta]\!]^E_D \equiv (\lambda b.(Xb))[X{:=}\lambda a.(Xaa')]\\
\equiv \lambda b.((\lambda a.(Xaa'))b)\\
=_{\alpha\beta} \lambda b.(Xba')
\end{array}$$

$$[\![r]\!]^E \equiv \lambda b.(Xba')$$

Take $\theta = [X{:=}a]$. Then $[\![\theta]\!]^E_D(X) \equiv \lambda a.a$, and

$$\begin{array}{rl}
[\![r]\!]^D[\![\theta]\!]^E_D \equiv (\lambda b.(Xb))[X{:=}\lambda a.a]\\
\equiv \lambda b.((\lambda a.a)b)\\
=_{\alpha\beta} \lambda b.b
\end{array}$$

$$[\![r\theta]\!]^E \equiv [\![[b]b]\!]^E \equiv \lambda b.b$$

Finally, take $\theta = [X{:=}a']$. Then $[\![\theta]\!]^E_D(X) \equiv \lambda a.a'$, and

$$\begin{array}{rl}
[\![r]\!]^D[\![\theta]\!]^E_D \equiv (\lambda b.(Xb))[X{:=}\lambda a.a']\\
\equiv \lambda b.((\lambda a.a')b)\\
=_{\alpha\beta} \lambda b.a'
\end{array}$$

$$[\![r\theta]\!]^E \equiv [\![[b]a']\!]^E \equiv \lambda b.a'$$

Now take $D = []$ and $E = [a, a', b]$, so that the condition of Theorem 40 fails. It is a fact that $[\![r]\!]^D \equiv \lambda b.X$. Take $\theta = id$. Then $[\![\theta]\!]^E_D(X) = Xaa'$ and

$$\begin{array}{rl}
[\![r]\!]^D[\![\theta]\!]^E_D \equiv ((\lambda b.X)[X{:=}Xaa']\\
\equiv \lambda b.(Xaa')
\end{array}$$

$$[\![r]\!]^E \equiv \lambda b.(Xba')$$

3.2 Soundness of the translation

In Theorem 43 we show that derivable equality is preserved by the translation, provided that the indexing vector is 'large enough'. We also verify that the axioms in ULAME translate to valid $\alpha\beta\eta$-equivalences between λ-terms (Theorem 45).

Lemma 42. *If Π is a derivation of $\mathsf{T} \vdash r = s$ and $\mathsf{T} \subseteq \mathsf{T}'$ then there exists a derivation Π' of $\mathsf{T}' \vdash r = s$, and $atoms(\Pi) = atoms(\Pi')$.*

Similarly, if Λ is a derivation of $\mathsf{L} \vdash g = h$ and $\mathsf{L} \subseteq \mathsf{L}'$ then there exists a derivation Λ' of $\mathsf{L}' \vdash g = h$, and $atoms(\Lambda) = atoms(\Lambda')$.

Proof. Since a derivation that mentions only axioms in T (or L) is also one that mentions only axioms in T' (or L'). □

Theorem 43. *Suppose Π is a derivation of $\mathsf{T} \vdash r = s$. Suppose $atoms(\Pi) \subseteq E$. Then there is a derivation $[\![\Pi]\!]^E$ of $[\![\mathsf{T}]\!] \vdash [\![r]\!]^E = [\![s]\!]^E$.*

Proof. By induction on Π.
- $(\mathbf{Nref})$, $(\mathbf{Ntran})$, $(\mathbf{Napp})$, and $(\mathbf{N}\lambda)$ translate to $(\mathbf{Lref})$, $(\mathbf{Ltran})$, $(\mathbf{Lapp})$, and $(\mathbf{L}\lambda)$ respectively.
- The case $(\mathbf{N}\alpha)$. Suppose $a, b \notin fa(r)$. Then $a, b \notin fa([\![r]\!]^E)$ by Lemma 39. By $(\mathbf{L}\alpha)$, $g =_\alpha (a\ b)\cdot g$. We use Lemma 42.
- The case $(\mathbf{Nax}^{\pi,\theta}_{\mathbf{r=s}})$ for $(r = s) \in \mathsf{T}$.
  Let D be the vector of atoms containing $atoms(r) \cup atoms(s)$ chosen in Definition 35.
  We need to show that $[\![\mathsf{T}]\!] \vdash [\![\pi \cdot (r\theta)]\!]^E = [\![\pi \cdot (s\theta)]\!]^E$. First, we note the following:

$$\begin{array}{rll}
[\![\pi \cdot (r\theta)]\!]^E \equiv \pi \cdot [\![r\theta]\!]^E & & \text{Lemma 38}\\
=_{\alpha\beta} (\lambda E.[\![r\theta]\!]^E)\pi \cdot E & & \text{Lemma 37}\\
=_{\alpha\beta} (\lambda E.([\![r]\!]^D[\![\theta]\!]^E_D))\pi \cdot E & & \text{Theorem 40}
\end{array}$$

  The use of Lemma 37 is valid since $nontriv(\pi) \subseteq E$ by assumption. The use of Theorem 40 is valid since $atoms(r) \subseteq D$ by assumption.
  By the same reasoning for s we note that

$$[\![\pi \cdot (s\theta)]\!]^E =_{\alpha\beta} (\lambda E.([\![s]\!]^D[\![\theta]\!]^E_D))\pi \cdot E.$$

  We use Lemma 42 and $(\mathbf{Lax}^{[\![\theta]\!]^{\mathbf{E}}_{\mathbf{D}}}_{[\![\mathbf{r=s}]\!]_{\mathbf{D}}})$. □

Example 44. Suppose $p(X) = comb$ and suppose $a, a' \in comb$ and $b \notin comb$. Consider term-formers A and E.

Consider a theory F with one axiom $\mathsf{E}[a]\mathsf{A}[b]X = \mathsf{E}[a]X$. Then $[\![\mathsf{F}]\!] = \{\mathsf{E}\lambda a.\mathsf{A}\lambda b.(Xa) = \mathsf{E}\lambda a.(Xa)\}$.

There is a trivial derivation of $\mathsf{F} \vdash \mathsf{E}[a]\mathsf{A}[b]X = \mathsf{E}[a]X$. Take $E = [a', a, b]$. It is not hard to construct a derivation of $[\![\mathsf{F}]\!] \vdash \mathsf{E}\lambda a.\mathsf{A}\lambda b.(Xa'a) = \mathsf{E}\lambda a.(Xa'a)$, by instantiating X to Xa' in $[\![\mathsf{F}]\!]$.

Theorem 45. *Suppose $(r = s) \in$ ULAME. For every π and θ, $[\![\pi \cdot (r\theta)]\!]^D =_{\alpha\beta\eta} [\![\pi \cdot (s\theta)]\!]^D$.*

Proof. We consider just two axioms of ULAME:
- Axiom $(\mathbf{N}\beta\mathbf{var})$. Since $(\lambda a.a)g =_{\alpha\beta\eta} g$ and $(\lambda b.b)g =_{\alpha\beta\eta} g$ always, by $(\mathbf{L}\beta\mathbf{var})$.
- Axiom $(\mathbf{N}\beta\#)$. The unknown X' in $(\mathbf{N}\beta\#)$ has permission set $comb \setminus \{a\}$. By Definition 10 $a \notin fa(\theta(X'))$. By Lemma 39 also $a \notin fa([\![\theta(X')]\!]^D)$. The result follows using $(\mathbf{L}\beta\#)$. □

4. Translating back

In Section 3 we translated nominal algebra theories to λ-theories. Now, we translate λ-theories to nominal algebra theories. As already discussed our translation identifies λ-abstraction with nominal atoms abstraction. Accordingly, it generates nominal algebra theories that include ULAME (Definition 14).

4.1 The translation $(\!|\text{-}|\!)^D$, and soundness

We define the inverse translation in Definition 47, then prove a compositionality result (Lemma 49). This resembles Theorem 40 but it is technically simpler, and holds for *all* vectors E. Our λ-terms have no capturing substitution, so this makes sense; the

vector in $(\!|\text{-}|\!)^D$ is just there to obtain inverse properties with respect to $[\![\text{-}]\!]^D$, as we shall see later in Subsections 4.3 and 4.4.

We prove a soundness result for the translation (Theorem 51). This resembles Theorem 43 but again, it is valid for all vectors.

Definition 46. Suppose $D = [d_1, \ldots, d_n]$. Continuing Definition 32 write $[D]r$ as shorthand for $[d_1] \ldots [d_n]r$ (a list of atoms-abstractions).

Definition 47. If g is a λ-term and D is a vector of atoms, define a nominal term $(\!|g|\!)^D$ by:

$$(\!|a|\!)^D \equiv a \qquad (\!|\mathsf{f}|\!)^D \equiv \mathsf{f} \qquad (\!|X|\!)^D \equiv [D \cap p(X)]X$$
$$(\!|\lambda a.g|\!)^D \equiv [a](\!|g|\!)^D) \qquad (\!|g'g|\!)^D \equiv (\!|g'|\!)^D (\!|g|\!)^D$$

Call σ **good** when $fa(\sigma(X)) \subseteq p(X)$ for all X. Extend the translation to good substitutions, equalities, and theories by:

$$(\!|\sigma|\!)^E_D(X) \equiv (\!|\sigma(X)|\!)^E(D \cap p(X)) \quad (\sigma \text{ good})$$
$$(\!|g = h|\!)^D \equiv ((\!|g|\!)^D = (\!|h|\!)^D)$$
$$(\!|\mathsf{L}|\!) = \{(\!|g = h|\!)^D \mid g = h \in \mathsf{T}\} \cup \mathsf{ULAME}$$

We take $(\!|\sigma|\!)^E_D$ to be undefined if σ is not good. The condition $fa(\sigma(X)) \subseteq p(X)$ is sufficient to ensure that $fa((\!|\sigma|\!)^E_D) \subseteq p(X)$ always, so that $(\!|\sigma|\!)^E_D$ is a substitution when it is defined (Definition 10). This condition is 'soft' in a sense discussed in Subsection 4.2.

In the definition of $(\!|\mathsf{L}|\!)$, D is $atoms(g) \cup atoms(h)$ in the order chosen in Definition 34.

Note that $(\!|\mathsf{L}|\!)$ is the translation of the axioms in L, *plus* ULAME; intuitively, ULAME represents the rules $(\mathbf{L}\beta\mathbf{var})$, $(\mathbf{L}\beta\#)$, $(\mathbf{L}\beta\mathbf{id})$, $(\mathbf{L}\beta\lambda)$, $(\mathbf{L}\beta\mathbf{app})$, and $(\mathbf{L}\eta)$.

Lemma 48. *$fa((\!|g|\!)^D) \subseteq fa(g)$.*

Proof. By induction on g. □

Lemma 49. *Suppose $fa(\sigma(X)) \cap D = []$ for every $X \in fV(g)$ and suppose σ is good. Then $\mathsf{ULAME} \vdash (\!|g\sigma|\!)^E_D = (\!|g|\!)^D (\!|\sigma|\!)^E_D$.*

Proof. By induction on g. We consider one case:

- The case of X. Write $S = p(X)$. We reason as follows:

$$\begin{array}{lll} \mathsf{ULAME} \vdash (\!|X|\!)^D (\!|\sigma|\!)^E_D \equiv ([D \cap S]X)(\!|\sigma|\!)^E_D & & \text{Definition 47} \\ \equiv [D \cap S](\!|\sigma|\!)^E_D(X) & & \text{Definition 11} \\ \equiv [D \cap S](\!|\sigma(X)|\!)^E(D \cap S)) & & \text{Definition 47} \\ = (\!|\sigma(X)|\!)^E & & (\mathbf{N}\eta) \end{array}$$

  The final step is valid, since $fa(\sigma(X)) \cap D = \varnothing$ by assumption and therefore by Lemma 48 also $fa((\!|\sigma(X)|\!)^E) \cap (D \cap S) = \varnothing$. □

Definition 50. Call a derivation Λ **good** when σ is good for every $(\mathbf{ax}^{\sigma}_{\mathbf{g=h}})$ in Λ.

Theorem 51. *Suppose Λ is a good derivation of $\mathsf{L} \vdash g = h$. Then for any E there is a derivation $(\!|\Lambda|\!)^E$ of $(\!|\mathsf{L}|\!) \vdash (\!|g|\!)^E = (\!|h|\!)^E$.*

Proof. We work by induction on Λ:

- $(\mathbf{Lrefl})$, $(\mathbf{Ltran})$, $(\mathbf{Lapp})$, and $(\mathbf{L}\lambda)$ translate to $(\mathbf{Nrefl})$, $(\mathbf{Ntran})$, $(\mathbf{Napp})$, and $(\mathbf{Nabs})$ respectively.
- The case $(\mathbf{Lax}^{\sigma}_{\mathbf{g=h}})$. We want that $(\!|\mathsf{L}|\!) \vdash (\!|g\sigma|\!)^E = (\!|h\sigma|\!)^E$. By Lemma 49 $(\!|\mathsf{L}|\!) \vdash (\!|g\sigma|\!)^E = (\!|g|\!)^{[]} (\!|\sigma|\!)^E_{[]}$ and $(\!|\mathsf{L}|\!) \vdash (\!|h\sigma|\!)^E = (\!|h|\!)^{[]} (\!|\sigma|\!)^E_{[]}$ (here $[]$ is the empty list of atoms). Also by $(\mathbf{Nax}^{id, (\!|\sigma|\!)^{\mathbf{E}}_{[]}}_{(\!|\mathbf{g=h}|\!)^{[]}})$ we have $(\!|\mathsf{L}|\!) \vdash id \cdot ((\!|g|\!)^E (\!|\sigma|\!)^E_{[]}) = id \cdot ((\!|h|\!)^{[]} (\!|\sigma|\!)^E_{[]})$. The result follows using $(\mathbf{Ltran})$ and Lemma 36.
- The case $(\mathbf{L}\beta\#)$. There are two cases:
  - $\mathsf{L} \vdash (\lambda a.h)g = h$ where $a \notin fa(h)$, and a is the atom we chose to use in the axiom $(\mathbf{N}\beta\#)$ of ULAME (Definition 14).
  - $\mathsf{L} \vdash (\lambda a'.h)g = h$ where $a' \notin fa(h)$, and a' is different from the atom a we chose to use in axiom $(\mathbf{N}\beta\#)$.

  We consider only the second case; the first case is similar.
  Let π be any permutation such that $\pi(a') = a$ and such that $\pi \cdot (fa(h) \cup fa(g)) \subseteq comb$ (that is, π maps a' to a and maps the atoms in $fa(h)$ to be inside $comb$, and the atoms in $fa(g)$ to be inside $comb$).
  Let θ be the substitution mapping X' to $\pi^{-1} \cdot ((\!|h|\!)^E)$ and X to $\pi^{-1} \cdot ((\!|g|\!)^E)$ and every other Y to $id \cdot Y$.
  We consider $(\mathbf{Nax}^{\pi,\theta}_{([\mathbf{a}]\mathbf{X}')\mathbf{X}=\mathbf{X}'})$, where $(([a]X')X = X') = (\mathbf{N}\beta\#)$. We verify that this gives the right result:
  By Lemmas 20 and 48 $fa(\pi \cdot ((\!|h|\!)^E)) \subseteq comb \backslash \{a\}$ and $fa(\pi \cdot ((\!|g|\!)^E)) \subseteq comb$. Therefore, θ is a substitution (Definition 10).
  Also:

$$\begin{array}{lll} \pi \cdot ((([a]X')X)\theta) \equiv \pi \cdot ([a]\pi^{-1} \cdot ((\!|h|\!)^E))(\pi^{-1} \cdot ((\!|g|\!)^E)) & & \text{Definition 11} \\ \equiv ([a'](\!|h|\!)^E))(\!|g|\!)^E) & & \text{Def. 8, Lem. 36} \\ \equiv (\!|([a']h)g|\!)^E & & \text{Definition 47} \end{array}$$

  Similarly, $\pi \cdot (X'\theta) \equiv (\!|h|\!)^E$ as required.
- The case of $(\mathbf{L}\eta)$ is similar (we translate to an instance of $(\mathbf{Nax})$ for $(\mathbf{N}\eta)$, and use Lemma 48).
- $(\mathbf{L}\beta\mathbf{var})$, $(\mathbf{L}\beta\lambda)$, $(\mathbf{L}\beta\mathbf{app})$ translate to instances of $(\mathbf{Nax})$ for $(\mathbf{N}\beta\mathbf{var})$, $(\mathbf{N}\beta\lambda)$, and $(\mathbf{N}\beta\mathbf{app})$ respectively. □

Example 52. Suppose $p(X) = comb$ and suppose $a, a' \in comb$ and $b \notin comb$. Consider term-formers A and E.
Consider a theory G with one axiom $\mathsf{E}\lambda a.\mathsf{A}\lambda b.(Xa) = \mathsf{E}\lambda a.(Xa)$. Then $(\!|\mathsf{G}|\!) = \{\mathsf{E}[a]\mathsf{A}[b](([a]X)a) = \mathsf{E}[a](([a]X)a)\}$.
There is a trivial derivation of $\mathsf{G} \vdash \mathsf{E}\lambda a.\mathsf{A}\lambda b.(Xa) = \mathsf{E}\lambda a.(Xa)$. Take $E = []$. It is not hard to construct a derivation of $(\!|\mathsf{G}|\!) \vdash \mathsf{E}[a]\mathsf{A}[b](Xa) = \mathsf{E}[a](Xa)$, by instantiating X to Xa in $(\!|\mathsf{G}|\!)$ and using $(\mathbf{N}\beta\mathbf{id})$.

4.2 Making derivations good

Theorem 51 uses a condition that Λ is 'good' (Definition 50). But what if we are given a Λ that is not? Lemmas 53 and 55 give senses in which we need not worry about this.

Lemma 53. *Suppose Π is a derivation of $\mathsf{T} \vdash r = s$. Suppose $atoms(\Pi) \subseteq E$. Then $[\![\Pi]\!]^E$ from Theorem 43 is a good derivation of $[\![\mathsf{T}]\!]^E \vdash [\![r]\!]^E = [\![s]\!]^E$.*

Proof. Examining the construction of $[\![\Pi]\!]^E$ in the proof of Theorem 43, it suffices to check that $[\![\theta]\!]^E_D$ is good for every θ in Π. Expanding definitions, we must check that $fa(\lambda D{\cap}p(X).[\![\theta(X)]\!]^E) \subseteq p(X)$ always. This is by Lemma 39 (with $r \equiv id \cdot X$). □

Definition 54. Call a bijection on unknowns a **renaming**. ρ will range over renamings.
Each ρ is also a substitution (Definition 25). Write $g\rho$ for ρ acting on g as a substitution. Write $\mathsf{L}\rho$ for the set $\{g\rho = h\rho \mid (g = h) \in \mathsf{L}\}$. Write $(\rho^{-1}\Lambda)$ for the tree obtained by mapping every $(\mathbf{ax}^{\sigma}_{\mathbf{g=h}})$ to $(\mathbf{ax}^{\rho^{-1} \circ \sigma}_{\mathbf{g}\rho=\mathbf{h}\rho})$.

It is a fact that if Λ is a derivation of $\mathsf{L} \vdash g = h$ then $(\rho^{-1}\Lambda)$ is a derivation of $\mathsf{L}\rho \vdash g = h$.

Lemma 55. *Suppose Λ is a derivation of $\mathsf{L} \vdash g = h$. Then there exists a ρ such that $(\rho^{-1}\Lambda)$ is a good derivation of $\mathsf{L}\rho \vdash g = h$.*

Proof. Let A be the set of all atoms mentioned in Λ. It suffices to choose ρ such that for each X mentioned in Λ, $A \subseteq p(\rho(X))$. □

4.3 The translations ⟦-⟧ and (|-|) are inverse on theories

Our main result is Theorem 62. This matches λ-theories with theories in nominal algebra that include ULAME.

Right from [32] the intuition of nominal terms and nominal algebra has been that this is a weaker syntax and logic than λ-terms and λ-theories; something that drops $\beta(\eta)$–equivalence, but retains just enough power to express binding. Theorem 62 is a way to make this intuition formal, by identifying arbitrary theories over λ-terms with nominal algebra theories over ULAME.

Lemma 56. $\mathsf{ULAME} \vdash ([a]r)a = r$.
$b \not\in fa(r)$ implies $\mathsf{ULAME} \vdash ([a]r)b = (a\ b) \cdot r$.
If $nontriv(\pi) \cap fa(r) \subseteq D$ then $\mathsf{ULAME} \vdash ([D]r)(\pi \cdot D) = \pi \cdot r$.

Proof. For the first two parts, we reason as follows:

$$\begin{array}{ll} \mathsf{ULAME} \vdash ([a]r)a = r & (\mathbf{N}\beta\mathbf{id}) \\ \mathsf{ULAME} \vdash ([a]r)b = ([b](a\ b) \cdot r)b & (\mathbf{N}\alpha),\ a,b \not\in fa([a]r) \\ \qquad = (a\ b) \cdot r & (\mathbf{N}\beta\mathbf{id}) \end{array}$$

The proof of the third part is like the proof of the third part of Lemma 37. □

Lemma 57. *If $atoms(r) \subseteq D$ then* $\mathsf{ULAME} \vdash (|[\![r]\!]^D|)^D = r$. *As a corollary, if $atoms(r) \subseteq D$ and* $\mathsf{ULAME} \subseteq \mathsf{T}$ *then* $\mathsf{T} \vdash (|[\![r]\!]^D|)^D = r$.

Proof. The corollary follows by Lemma 42. We prove the first part by induction on r:

- The case $\pi \cdot X$. Write $S = p(X)$. We reason as follows:

$$\begin{array}{lll} \mathsf{ULAME} \vdash (|[\![\pi \cdot X]\!]^D|)^D \equiv (|X(\pi \cdot (D \cap S))|)^D & & \text{Definition 35} \\ \qquad \equiv ([D \cap S]X)(\pi \cdot (D \cap S)) & & \text{Definition 47} \\ \qquad = \pi \cdot X & & \text{Lemma 56} \end{array}$$

  The use of Lemma 56 here is valid since $nontriv(\pi) \subseteq D$ by assumption and $S \subseteq D$, and it follows by Definition 9 that $nontriv(\pi) \cap fa(X) \subseteq D \cap S$. □

Lemma 58. $[\![(|g|)^D]\!]^D =_\eta g$.
As a corollary, $\mathsf{L} \vdash [\![(|g|)^D]\!]^D = g$ *always.*

Proof. The corollary follows by Lemma 42. We prove the first part by induction on g. We consider one case:

- The case X. Write $S = p(X)$. We reason as follows:

$$\begin{array}{lll} [\![(|X|)^D]\!]^D \equiv [\![[D \cap S]X]\!]^D & & \text{Definition 47} \\ \qquad \equiv \lambda(D \cap S).(X(D \cap S)) & & \text{Definition 35} \\ \qquad =_\eta X & & (\mathbf{L}\eta) \end{array}$$

□

Definition 59. Suppose T and T' are two theories. Write $\mathsf{T} \dashv\vdash \mathsf{T}'$ when for all g and h, $\mathsf{T} \vdash g = h$ if and only if $\mathsf{T}' \vdash g = h$.
Similarly for L and L'.

Lemma 60. $\mathsf{T} \dashv\vdash \mathsf{T}'$ *if and only if* $\mathsf{T} \vdash g' = h'$ *for every* $(g' = h') \in \mathsf{T}'$ *and also* $\mathsf{T}' \vdash g = h$ *for every* $(g = h) \in \mathsf{T}$.
Similarly for L *and* L'.

Proof. The left-to-right implication is immediate. The right-to-left implication is by a routine induction on derivations. □

Lemma 61. *We observe the following set inclusions:*

$$atoms(r) \subseteq atoms([\![r]\!]^D) \subseteq atoms(r) \cup D$$
$$atoms(g) \subseteq atoms((|g|)^D) \subseteq atoms(g) \cup D$$

Proof. By routine inductions on r and g. □

Theorem 62. *If* $\mathsf{ULAME} \subseteq \mathsf{T}$ *then* $(|[\![\mathsf{T}]\!]|) \dashv\vdash \mathsf{T}$.
$[\![(|\mathsf{L}|)]\!] \dashv\vdash \mathsf{L}$ *always.*

Proof. We use Lemma 60.

Suppose $(r = s) \in \mathsf{T}$. Let D be the vector of atoms chosen for $atoms(r) \cup atoms(s)$ in Definition 34. By Lemma 61 $atoms([\![r]\!]^D) \cup atoms([\![s]\!]^D) = atoms(D)$. Therefore, $(|[\![r]\!]^D|)^D = (|[\![s]\!]^D|)^D \in (|[\![\mathsf{T}]\!]|)$. By assumption $atoms(r) \subseteq D$ and $atoms(s) \subseteq D$. The result follows by $(\mathbf{Ntran})$ and by Lemma 57.

Conversely, suppose $(r = s) \in (|[\![\mathsf{T}]\!]|)$. From Definition 47 and Lemma 61 there are two cases:

- $(r = s) \in \mathsf{ULAME}$.
- $(r = s) = ((|[\![r']\!]^D|)^D = (|[\![s']\!]^D|)^D)$ for some $(r' = s') \in \mathsf{T}$. We use Lemma 57 as before.

Suppose $(g = h) \in \mathsf{L}$. Let D be the vector of atoms chosen for $atoms(g) \cup atoms(h)$ in Definition 34. By Lemma 61 $atoms((|g|)^D) \cup atoms((|h|)^D) = atoms(D)$. Therefore, $([\![(|g|)^D]\!]^D = [\![(|h|)^D]\!]^D) \in [\![(|\mathsf{L}|)]\!]$. We use $(\mathbf{Ltran})$ and Lemma 58.

Suppose $(g = h) \in [\![(|\mathsf{L}|)]\!]$. There are two possibilities:

- $(g = h) = ([\![r]\!]^D = [\![s]\!]^D)$ for some $(r = s) \in \mathsf{ULAME}$. By Theorem 45 and Lemma 42, $\mathsf{L} \vdash g = h$.
- $(g = h) = ([\![(|g'|)^D]\!]^D = [\![(|h'|)^D]\!]^D)$ for some $(g' = h') \in \mathsf{L}$. We use Lemma 58 as before. □

4.4 The translations ⟦-⟧^{D} and (|-|)^D are inverse on derivations, if D is sufficiently large

We now come to our main result; Theorem 67. Intuitively we can read this as follows: we can translate between nominal algebra theories (with ULAME) and λ-theories. In both directions the translation uses a finite vector of atoms E. For derivability of equality to be preserved in both directions, this vector must be 'large enough'.

Thus, there is no fixed 1-1 correspondence between nominal terms and λ-terms, but we can translate between theories (Subsection 4.3) and *for any given derivation of an equality*, if we take a large enough indexing vector, then we can also translate that derivation accurately between the two theories. There may always be another derivation of a different equality, or even of the same equality, that requires a different (possibly longer, possibly shorter) vector.

Note that we have not attempted to exhibit a 'minimal' vector such that the translation works; this is future work. Perhaps the notion of the *capturable atoms* of a nominal term may be relevant here; it was used to measure the minimal raising required to translate between nominal and higher-order unification [7].

Lemma 63. *If* Π *is a derivation of* $\mathsf{T} \vdash r = s$ *then there exists a derivation* $\Pi\theta$ *of* $\mathsf{T} \vdash r\theta = s\theta$. *Furthermore,* $atoms(\Pi) \subseteq atoms(\Pi\theta) \subseteq atoms(\Pi) \cup \bigcup\{fa(\theta(X)) \mid X \text{ occurs in } \Pi\}$.

Proof. We apply θ to every term in Π to obtain $\Pi\theta$ and verify by induction that $\Pi\theta$ is a derivation of $\mathsf{T} \vdash r\theta = s\theta$. □

Corollary 64. *Suppose* Π *is a derivation of* $\mathsf{T} \vdash (|g|)^D = (|h|)^D$. *Suppose* $atoms(\mathsf{ULAME}) \subseteq D \subseteq E$. *Then there is a derivation* Π' *of* $\mathsf{T} \vdash (|g|)^E = (|h|)^E$, *and* $atoms(\Pi) \subseteq atoms(\Pi') \subseteq atoms(\Pi) \cup E$.

Proof. By Lemma 63 $\Pi(|id|)^E_D$ is a derivation of $\mathsf{T} \vdash (|g|)^D(|id|)^E_D = (|h|)^D(|id|)^E_D$, and $atoms(\Pi) \subseteq atoms(\Pi(|id|)^E_D) \subseteq atoms(\Pi) \cup D$. By Lemma 49 $\mathsf{T} \vdash (|g|)^D(|id|)^E_D = (|g|)^E$ and we see from the proof of Lemma 49 that the atoms in the derivation are in $atoms(g) \cup E \subseteq atoms(\Pi) \cup E$. Similarly for h. We use $(\mathbf{Ltran})$. □

Theorem 65. *Suppose* $\mathsf{ULAME} \subseteq \mathsf{T}$.

- *Suppose* Π *is a derivation of* $\mathsf{T} \vdash r = s$. *Then* $[\![\Pi]\!]^E$ *is a derivation of* $[\![\mathsf{T}]\!] \vdash [\![r]\!]^E = [\![s]\!]^E$ *for any* E *such that* $atoms(\Pi) \subseteq E$.
- *Suppose* $[\![\mathsf{T}]\!] \vdash [\![r]\!]^E = [\![s]\!]^E$ *and* $atoms(r) \cup atoms(s) \subseteq E$. *Then* $\mathsf{T} \vdash r = s$.
- *Suppose* Λ *is a derivation of* $\mathsf{L} \vdash g = h$. *There exists a renaming* ρ *(Definition 54) such that for any* E, $(|\mathsf{L}\rho|) \vdash (|g|)^E = (|h|)^E$.

- *Suppose* $(\!|\mathsf{L}\rho|\!) \vdash (\!|g|\!)^E = (\!|h|\!)^E$ *for some* ρ *and some* E. *Then* $\mathsf{L} \vdash g = h$.

Proof. - This is Theorem 43.
- Suppose $[\![\mathsf{T}]\!] \vdash [\![r]\!]^E = [\![s]\!]^E$ where $atoms(r) \cup atoms(s) \subseteq E$. By Lemma 53 and Theorem 51 $(\!|[\![\mathsf{T}]\!]|\!) \vdash (\!|[\![r]\!]^E|\!)^E = (\!|[\![s]\!]^E|\!)^E$. By Theorem 62 $(\!|[\![\mathsf{T}]\!]|\!) \dashv\vdash \mathsf{T}$. By Lemma 57 since $\mathsf{ULAME} \subseteq \mathsf{T}$ and $atoms(r) \subseteq E$, $\mathsf{T} \vdash (\!|[\![r]\!]^E|\!)^E = r$, and similarly for s. It follows by $(\mathbf{Ltran})$ that $\mathsf{T} \vdash r = s$ as required.
- By Lemma 55 there exists a ρ such that $(\rho^{-1}\Lambda)$ is a good derivation of $\mathsf{L}\rho \vdash g = h$. The result follows by Theorem 51.
- Suppose Π is a derivation of $(\!|\mathsf{L}\rho|\!) \vdash (\!|g|\!)^E = (\!|h|\!)^E$. Choose any E' such that $E \cup atoms(\Pi) \cup atoms(\mathsf{ULAME}) \subseteq E'$. By Corollary 64 there is a derivation Π' of $(\!|\mathsf{L}\rho|\!) \vdash (\!|g|\!)^{E'} = (\!|h|\!)^{E'}$ and $atoms(\Pi') \subseteq E'$. By Theorem 43 $[\![(\!|\mathsf{L}\rho|\!)]\!] \vdash [\![(\!|g|\!)^{E'}]\!]^{E'} = [\![(\!|h|\!)^{E'}]\!]^{E'}$. By Theorem 62 $[\![(\!|\mathsf{L}\rho|\!)]\!] \dashv\vdash \mathsf{L}\rho$. By Lemma 58, $\mathsf{L}\rho \vdash [\![(\!|g|\!)^{E'}]\!]^{E'} = g$, and similarly for h. It follows by $(\mathbf{Ltran})$ that $\mathsf{L}\rho \vdash g = h$. Since ρ is invertible, we can use Lemma 55 to verify that $\mathsf{L} \vdash g = h$ as required. □

Definition 66. Suppose $\Phi(D)$ is some assertion (in English) which depends on a vector D. Write "$\Phi(D)$ holds for sufficiently large D" to mean "there exists a D such that for all D', if $D \subseteq D'$ (Definition 31) then $\Phi(D')$".

Theorem 67. *Suppose* $\mathsf{ULAME} \subseteq \mathsf{T}$. *Then:*
- $\mathsf{T} \vdash r = s$ *if and only if* $[\![\mathsf{T}]\!] \vdash [\![r]\!]^E = [\![s]\!]^E$ *for sufficiently large* E.
- *For any* E, $\mathsf{L} \vdash g = h$ *if and only if there exists a* ρ *such that* $(\!|\mathsf{L}\rho|\!) \vdash (\!|g|\!)^E = (\!|h|\!)^E$.

Proof. Direct from Theorem 65. □

A stronger version of part 2 of Theorem 67 holds, where we echo part 1 and write "for sufficiently permissive ρ", meaning for "ρ such that $p(X)$ is sufficiently large for relevant X" (generalising Definition 66 to sets). Making this formal is for a longer paper.

5. Conclusions

Equational reasoning is simple, yet powerful. In this arena we have addressed the question "what is the connection between truth and derivability over nominal syntax, and truth and derivability over higher-order syntax".

No single translation offers itself — but for any given derivation we can find a sufficiently large indexing set to translate it between the two worlds. This makes sense, because nominal terms unknowns X intuitively represent 'any term, with a capturing substitution', and a term may mention finite but unboundedly many atoms. The translation also brings out some nice features of using (permissive) nominal terms syntax, compared to λ-terms syntax:

- Nominal terms are more elementary, in the sense that the primitive theory of equality over nominal terms (no axioms) has fewer axioms than the primitive theory of equality over λ-terms (which the results in this paper identify in a certain precise sense with the axioms of ULAME). In other words: to reason in permissive nominal algebra we do not need to include the behaviour captured by the axioms in ULAME if we do not want it, whereas to reason using λ-terms, we must include that behaviour whether we want it or not (perhaps minus $(\mathbf{L}\eta)$).[6]
- By design, nominal terms have a two-level variable structure which directly models the behaviour of the informal meta-level of mathematical discourse with its 'object-level variables' and 'meta-level variables' (see the example statements about first-order logic and λ-calculus in the Introduction). As we move under abstractions, it is not necessary to maintain a list of names for which we wish to permit capture; this list appears in this paper as the $D \cap p(X)$ in Definitions 35 and 47.

It is possible that logics based on nominal terms, of which permissive nominal algebra is one example, might offer convenient reasoning frameworks for formalising informal reasoning about logic and computation. It is a natural next step to study reasoning using nominal terms in more detail, and this includes considering logics with richer judgement-forms. We hope the techniques used here will be prototypical for such logics.

Choice of permissive nominal terms over nominal terms The nominal algebra presented in this paper is not identical to that presented in [14, 26]. We use a 'permissive' variant which we find simpler to define and manipulate. The connection with 'standard' nominal algebra will formally be developed in a later paper. We will, however, sketch why we prefer to use permissive nominal terms; for definitions and notation for nominal terms, see [32].

Nominal terms have no permissions sets. Instead there is a freshness context, usually written Δ. Δ is a finite set of assertions $a\#X$; these correspond in the notation of this paper with $a \notin p(X)$ ("a is not in the permissions set of X"). Thus the translation of Definition 35 would be indexed by Δ, so we might write it as $[\![r]\!]^D_\Delta$.

The difference is that we may need to change Δ, and there is no natural notion of $fa(r)$. Often, we must obtain fresh atoms; see for example the case $[a]r$ in Theorem 40. Using permissive nominal terms such fresh atoms are readily available by Lemma 18. In the theory of nominal terms this is not the case; for example Δ may be empty and so provide no fresh atoms at all. Therefore, we would have to extend Δ to a 'freshly extended' freshness context $\Delta' \supseteq \Delta$. It might be useful to sketch what Theorem 40 would become:

Suppose $atoms(r) \subseteq D$. *Then for any* Δ, *there exists some freshly extended* $\Delta' \supseteq \Delta$ *such that* $[\![r\theta]\!]^E_{\Delta'} =_{\alpha\beta} [\![r]\!]^D_{\Delta'}[\![\theta]\!]_{\Delta'}{}^E_D$.

There is nothing mathematically wrong with this, but it is more complex and this complexity propagates. Uses of the result may involve reasoning on and possible extension of the freshness context; this lengthens proofs, propagates to the statements of following results, and it introduces a non-trivial notion of state into our reasoning in the sense that the freshness contexts may get larger at each reasoning step.

The use of permissive nominal syntax avoids this and so gives us shorter and sweeter proofs.

Future work The algebras considered in this paper are unsorted and untyped. λ-terms are routinely given types; we could do the same for nominal terms. A 'function type' $\alpha \to \beta$ would just indicate a term binding of a variable of type α in a term of type β — but using axioms like those in ULAME we could also endow this with functional behaviour if we wished.

The encoding of λ-theories into nominal algebra theories suggests we might be able to apply nominal algebra theorems to study existing work on logic and computation phrased in universal algebra. We have particularly in mind Salibra's study of λ-theories. Nominal algebra satisfies some stronger properties than univeral algebra, notably a strong version of the HSP theorem [11] which factors also over atoms-abstraction. It may be possible to put some constructions of Salibra, which he carries out specifically for Lambda-Abstraction Algebras, into a nominal algebraic setting.

As discussed in Subsection 2.2, the proofs in this paper require η-equivalence. [16, 18] consider a non-extensional theory ULAM

[6] This point also motivates *higher-order patterns* [27]. See [6, 7, 23] for the connection with nominal terms.

(it lacks $(\mathbf{N}\eta)$). It is future work to see whether the proof-methods here can be adjusted to work without it. An overview of some denotations for logic over non-extensional λ-terms is [3].

It should be possible to use the translation of λ-theories into nominal algebra, to construct denotations for them in nominal sets or variants of nominal sets. Conversely, if a nominal algebra theory contains ULAME then it should be possible to give it a denotation in functions (probably a rather large one, involving all possible indexes over vectors of atoms).

It would be interesting to find a translation which is sound and complete for atoms-abstraction *without* ULAME: that is, can we interpret nominal atoms-abstraction in λ-syntax such that the image of the translation does *not* satisfy β-equivalence? It might then be possible to give denotations based on functions (as well as those based on nominal sets [17]) to nominal terms and (arbitrary) nominal algebra theories.

References

[1] Beatrice Amrhein. Birkhoff's HSP-theorem for cumulative logic programs. *Lecture Notes in Computer Science*, 798:24–36, 1994.

[2] Peter B. Andrews. *An introduction to mathematical logic and type theory: to truth through proof*. Academic Press, 1986.

[3] Christoph Benzmüller, Chad E. Brown, and Michael Kohlhase. Higher-order semantics and extensionality. *Journal of Symbolic Logic*, 69:1027–1088, 2004.

[4] S. Burris and H. Sankappanavar. *A Course in Universal Algebra*. Graduate texts in mathematics. Springer, 1981.

[5] James Cheney. Relating nominal and higher-order pattern unification. In *Proceedings of the 19th International Workshop on Unification (UNIF 2005)*, pages 104–119, 2005.

[6] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. Technical Report HW-MACS-TR-0062, Heriot-Watt University, 2009. Available online at gabbay.org.uk/papers/perntu-tr.pdf.

[7] Gilles Dowek, Murdoch J. Gabbay, and Dominic P. Mulligan. Permissive Nominal Terms and their Unification. In *CILC, 24th Italian Conference on Computational Logic*, 2009.

[8] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157, 2000.

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

[10] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, Cambridge, UK, 2000.

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

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

[13] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. Technical Report HW-MACS-TR-0045, Heriott-Watt, 2006.

[14] 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, 2007.

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

[16] 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.

[17] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. *Journal of Logic and Computation*, 2009. In press.

[18] Murdoch J. Gabbay and Aad Mathijssen. A nominal axiomatisation of the lambda-calculus. *Journal of Logic and Computation*, 2009. In press.

[19] Murdoch J. Gabbay and A. M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

[20] Jan Friso Groote, Aad Mathijssen, Michel Reniers, Yaroslav Usenko, and Muck van Weerdenburg. The formal specification language mCRL2. In *Methods for Modelling Software Systems (MMOSS)*, number 06351 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), 2007.

[21] Jan Friso Groote, Aad Mathijssen, Muck van Weerdenburg, and Yaroslav S. Usenko. From μCRL to mCRL2: motivation and outline. In *Proc. Workshop Algebraic Process Calculi: The First Twenty Five Years and Beyond*, BRICS NS-05-3, pages 126–131, 2005.

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

[23] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. In *Proceedings of RTA'08*, volume 5117 of *Lecture Notes in Computer Science*. Springer, 2008.

[24] Stefania Lusin and Antonino Salibra. The lattice of lambda theories. *Journal of Logic and Computation*, 14(3):373–394, 2004.

[25] Giulio Manzonetto and Antonino Salibra. Applying universal algebra to lambda calculus. *Journal of Logic and computation*, 2009. Online first.

[26] Aad Mathijssen. *Logical Calculi for Reasoning with Binding*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[27] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497 – 536, 1991.

[28] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14(4):321–358, 1992.

[29] Donald Monk. On the representation theory for cylindric algebras. *Pacific Journal of Mathematics*, 11(4):1447–1457, 1961.

[30] Antonino Salibra. On the algebraic models of lambda calculus. *Theoretical Computer Science*, 249(1):197–240, 2000.

[31] Li-Yang Tan. *Formalizing the meta-theory of $\mathcal{Q}_0$ in the calculus of inductive constructions*. PhD thesis, Henry Edwin Sever Graduate School of Washington University, 2006.

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