# A nominal axiomatisation of the lambda-calculus

Murdoch J. Gabbay[1,2] Aad Mathijssen[3]

Abstract

The lambda-calculus is fundamental in computer science. It resists an algebraic treatment because of capture-avoidance side-conditions.
Nominal algebra is a logic of equality designed for specifications involving binding. We axiomatise the lambda-calculus using nominal algebra, demonstrate how proofs with these axioms reflect the informal arguments on syntax, and we prove the axioms sound and complete. We consider both non-extensional and extensional versions (alpha-beta and alpha-beta-eta equivalence).
This connects the nominal approach to names and binding with the view of variables as a syntactic convenience for describing functions. The axiomatisation is finite, close to informal practice, and it fits into a context of other research such as nominal rewriting and nominal sets.

*Keywords:* lambda calculus, equational logic, nominal techniques

## Contents



## 1 Introduction

Functions are widely used in what we now call computer science; a development which can be traced back to Church [Chu41]. They are the basis of functional programming languages [Pau96,Tho96]; they also find application in logic [Bar77,Lei94], theorem-provers

[1] We thank an anonymous referee of a previous paper for comments which set us on the path to writing this one, the anonymous referees, and Chantal Berline for comments and support. Supported by grant RYC-2006-002131 at the Polytechnic University of Madrid.

[2] Homepage: http://www.gabbay.org.uk

[3] Email: a.h.j.mathijssen@tue.nl

[ABI+96,Pau89], rewriting [BN98], and more. Functions are a basic mathematical entity of computer science.

Functions have been studied in different ways. Models include Scott domain models, graph and filter models, game models, and more [CDHL84,KNO02,Ber00,Ber06]. There is much study using rewriting, starting with the proof of confluence of $\beta$-reduction (a particularly neat proof is in [Tak95]). There are also axiomatisations, including axioms by Andrews [And86], $\lambda$-algebras [Sel02], Salibra's lambda-abstraction algebras [Sal00], and (if one does not care about representing $\lambda$-abstraction, which we do) $\lambda$-lifting [Joh85].

This paper will axiomatise the $\lambda$-calculus using a recent logic by the authors: *nominal algebra* [GM06,Mat07,GM07,GM09]. Nominal algebra extends universal algebra [Coh65] with support for names and binding, while preserving much of its flavour and its good mathematical properties. Using nominal algebra to axiomatise the $\lambda$-calculus brings several benefits:

- Nominal algebra seems quite good at bringing us close to informal practice. It is easy to write down plausible axioms for the $\lambda$-calculus; they look just like well-known informal $\alpha\beta(\eta)$-equivalences. In this paper we shall consider two nominal algebra theories:
  - ULAM (Figure 1). This axiomatises the $\lambda$-calculus up to $\alpha\beta$-equivalence (Theorems 4.3 and 4.7).
  - ULAME (Figure 5). This axiomatises the $\lambda$-calculus up to $\alpha\beta\eta$-equivalence (Theorems 5.3 and 5.4).
- The sets of axioms ULAM and ULAME are *finite*. Figures 1 and 5 do not describe infinite schemes of axioms (unlike is the case for example in [Sal00] or [And86]; see Remark 2.13). This is because name-binding are handled by nominal algebra itself. [4]
- Nominal algebra is not a stand-alone logic; it is part of a body of research which includes nominal unification and nominal rewriting [UPG04,FG07b] (both of which have good computational properties) — and *nominal sets*:
- ULAM and ULAME are nominal algebra theories, and so they take models in nominal sets [GM07,GP01]. [5]

  The theory of these nominal sets models is investigated in [Gab09]. They have properties that sets models of universal algebra theories do not have. The notion of *variety* is richer, and nominal sets models are inherently *finite-dimensional* and exclude *infinite-dimensional* models. [6] Thus, using nominal techniques give us the option of working purely (nominal) algebraically in a finite-dimensional world, which is an option that universal algebra does not offer.

So we can write down axioms which look plausibly like they axiomatise functional abstraction, in an algebraic logic which looks plausibly like universal algebra and which is compatible with a broader body of research. But ... in what sense are the axioms correct? We can ask three questions:

[4] See [NPP08, Section 7] for an outline of how a similar idea can be implemented in a completely different kind of language.

[5] For the impatient: a nominal set is a set with a name-permutation action. Name-binding is formed by a construction very similar to taking $\alpha$-equivalence classes; we use the permutation action to 'rename' the name to be bound and taking an equivalence class of all the renamed variants.
Nominal sets can be considered as sheaves, or as algebraic structures. They were introduced in [GP01]; see that paper for full details.

[6] This is terminology from cylindric techniques (see [Sal00, Definition 7] or [HMT85]). Nominal sets terminology calls this *finitely-supported* and *infinitely-supported* respectively.

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

**Figure 1:** Axioms of ULAM

- If we quotient $\lambda$-terms by $\alpha\beta$-equivalence (respectively, $\alpha\beta\eta$-equivalence) do we obtain, in some natural way, a model of ULAM (respectively, ULAME)?
- Are the equalities which are valid in all models of ULAM, precisely the equalities described by that one set or is there something missing? (Is ULAM *complete* for $\alpha\beta$?) Similarly for ULAME.
- Does nominal algebra reasoning capture a useful fragment of the kind of reasoning steps we would like to represent?

In this paper we explore to what extent our two theories ULAM and ULAME capture 'the $\lambda$-calculus and its theory'. We will demonstrate that ULAM and ULAME are theories for the untyped $\lambda$-calculus, in the sense that they are sound and complete for $\lambda$-terms quotiented by $\alpha\beta$-equivalence and $\alpha\beta\eta$-equivalence respectively. We will demonstrate with examples how they can express informal reasoning as formal derivations.

*Nota bene:*
Nominal techniques were first applied to construct datatypes of syntax-with-binding [GP01] with good inductive reasoning principles. One datatype often used is $\lambda$-term (up to $\alpha$-equivalence). This paper is not another such study, like those in nominal sets [GP01], higher-order abstract syntax [PE88], de Bruijn terms [dB72], and so on — which are about collections of syntax trees.

*Overview of the paper*
We introduce nominal algebra in Section 2, with a syntax specialised to our application to the $\lambda$-calculus (a general treatment is elsewhere [Mat07]). In Section 3 we provide a brief formal treatment of the $\lambda$-calculus. In Section 4 we show that the nominal algebra theory ULAM is sound and complete with respect to $\alpha\beta$-equality. In Section 5 we show that the nominal algebra theory ULAME is sound and complete with respect to $\alpha\beta\eta$-equality. In the Conclusions (Section 6) we discuss related and future work. In Appendix A we include, for the reader's convenience, statements and proofs of some underlying properties of nominal algebra; these do not depend on ULAM or ULAME but they are relevant background, in much the same way as the definitions and results in Section 3 are relevant background.

## 2 Nominal algebra

In this section, we present the proof theory of nominal algebra. It consists of an equational logic on nominal terms, and has built-in support for binding, freshness and meta-variables.

### 2.1 *Nominal terms*

We define a syntax of nominal terms tailored to our $\lambda$-calculus application; general treatments are elsewhere [GM08a,Mat07].

**Definition 2.1** Fix the following disjoint sets:

- A countably infinite set of **atoms** $\mathbb{A}$. Atoms represent object-level variables.
  $a, b, c, \ldots$ will range over atoms. We use a **permutative convention** that $a, b, c, \ldots$ range over *distinct* atoms. Thus for example in $(\#\mathbf{ab})$ and $(\#\lambda\mathbf{b})$ from Figure 2, and in $(\mathbf{perm})$ from Figure 3, $a$ and $b$ represent two *distinct* atoms. [7]
- A countably infinite collection of **unknowns**. Unknowns represent meta-variables (as in 'take a term $t$'; $t$ is a meta-variable ranging over terms).
  $X, Y, Z, \ldots$ will range over distinct unknowns.
- a possibly infinite collection of **constant symbols**. $\mathsf{c}$ will range over constant symbols (we will never need to consider more than one at a time).

We set about constructing the machinery of nominal algebra.

**Definition 2.2** A **permutation** $\pi$ of atoms is a bijection on atoms with **finite support**, which means that the set $\mathsf{supp}(\pi)$, defined by $\{a \mid \pi(a) \neq a\}$, is finite. In words: For 'most' atoms $\pi$ is the identity.

The following notation will prove convenient:

- Write $id$ for the **identity** permutation on atoms, $\pi^{-1}$ for the **inverse** of $\pi$, and $\pi \circ \pi'$ for the **composition** of $\pi$ and $\pi'$, i.e. $(\pi \circ \pi')(a) = \pi(\pi'(a))$.
- Write $(a\ b)$ for the permutation that **swaps** $a$ and $b$, i.e. $(a\ b)(a) = b$, $(a\ b)(b) = a$, and $(a\ b)(c) = c$. We may omit $\circ$ between swappings, writing $(a\ b) \circ (b\ c)$ as $(a\ b)(b\ c)$.

**Definition 2.3** **Terms** $t, u, v$ are inductively defined by:

$$t \ ::= \ a \mid \pi \cdot X \mid \lambda a.t \mid tt \mid \mathsf{c}$$

We will use the following conventions:

- Application is left-associative, so for example '$tuv$' means '$(tu)v$'.
- Abstraction extends as far to the right as possible, so for example '$\lambda a.tu$' means '$\lambda a.(tu)$'.
- We may write $id \cdot X$ just as $X$. However, note that '$X$' is an unknown and *not* a term — '$id \cdot X$' is a term.
- We write $\equiv$ for **syntactic identity**. That is, '$t \equiv u$' means '$t$ and $u$ denote the same term'.

A typed syntax is possible; see [FG07a]. Types would cause no essential difficulties for the results to follow.

We now give some basic definitions; of the atoms-permutation action and of the capturing substitution action, which are characteristic of nominal terms [UPG04].

[7] Besides being useful in what follows, this models common practice: if we ask the reader to 'consider two variable symbols $x$ and $y$' then we have no control over, for example, their handwriting, and thus over the symbols which they actually commit to the page. What matters is that the two variable symbols are *different*.

**Definition 2.4** Define the set $atoms(t)$ of atoms that occur anywhere in $t$ inductively by:

$$atoms(a) = \{a\} \qquad atoms(\pi \cdot X) = \mathsf{supp}(\pi) \qquad atoms(\lambda a.t) = atoms(t) \cup \{a\}$$
$$atoms(t't) = atoms(t') \cup atoms(t) \qquad atoms(\mathsf{c}) = \emptyset$$

We also write $atoms(t_1, \ldots, t_n)$ as a shorthand for $atoms(t_1) \cup \cdots \cup atoms(t_n)$.

For example,

$$atoms(\lambda a.a) = \{a\} \qquad atoms((b\ a) \cdot X) = \{b, a\} \qquad atoms(\mathsf{f}(a, a)) = \{a\}.$$

**Definition 2.5** Define a **permutation action** $\pi \cdot t$ by:

$$\pi \cdot a \equiv \pi(a) \qquad \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \qquad \pi \cdot \lambda a.t \equiv \lambda(\pi(a)).(\pi \cdot t)$$
$$\pi \cdot (t't) \equiv (\pi \cdot t')(\pi \cdot t) \qquad \pi \cdot \mathsf{c} \equiv \mathsf{c}$$

In the clause for $\lambda$, $\pi$ acts also on the '$a$'. For example $(a\ b) \cdot \lambda a.X \equiv \lambda b.(a\ b) \cdot X$. In the clause for $\pi' \cdot X$, $\pi' \cdot X$ is a term (recall that '$X$', on its own, is not a term; it must always be paired with a permutation, even if it is $id$).

**Definition 2.6** A **substitution** $\sigma$ is a function from unknowns to terms.

**Definition 2.7** Define a **substitution action** $t\sigma$ by:

$$a\sigma \equiv a \qquad (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) \qquad (\lambda a.t)\sigma \equiv \lambda a.(t\sigma) \qquad (t't)\sigma \equiv (t'\sigma)(t\sigma) \qquad \mathsf{c}\sigma \equiv \mathsf{c}$$

Note that substitution does not avoid capture; note also that when $\sigma$ encounters $\pi \cdot X$, the permutation $\pi$ is applied to $\sigma(X)$. For example if $\sigma(X) \equiv a$ then:

$$\begin{aligned}(\lambda a.X)\sigma &\equiv \lambda a.(X\sigma)\\ &\equiv \lambda a.\sigma(X)\\ &\equiv \lambda a.a\end{aligned} \qquad \begin{aligned}(\lambda b.(a\ b) \cdot X)\sigma &\equiv \lambda b.(((a\ b) \cdot X)\sigma)\\ &\equiv \lambda b.(a\ b) \cdot \sigma(X)\\ &\equiv \lambda b.(a\ b) \cdot a\\ &\equiv \lambda b.b\end{aligned}$$

### *2.2 Freshness, equality, axioms, theories*

**Definition 2.8** A **freshness** is a pair $a\#t$ of an atom and a term. Call a freshness of the form $a\#X$ (so $t \equiv X$) **primitive**. Write $\Delta$ and $\nabla$ for (finite, and possibly empty) sets of *primitive* freshnesses and call them **freshness contexts**.

We may drop set brackets in freshness contexts. For example we may write $a\#X, b\#Y$ for $\{a\#X, b\#Y\}$.

**Definition 2.9** Define **derivability on freshnesses** by the rules in Figure 2. In this figure, $a$ and $b$ permutatively range over atoms, $t$ and $t'$ range over nominal terms, $\pi$ over permutations of atoms, $X$ over unknowns, and $\mathsf{c}$ over constants.

Write $\Delta \vdash a\#t$ when a derivation of $a\#t$ exists using these rules such that the assumptions are elements of $\Delta$. We usually write $\emptyset \vdash a\#t$ as $\vdash a\#t$.

$$\frac{}{a\#b}\,(\#\mathbf{ab}) \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}\,(\#\mathbf{X}) \qquad \frac{}{a\#\lambda a.t}\,(\#\lambda\mathbf{a}) \qquad \frac{a\#t}{a\#\lambda b.t}\,(\#\lambda\mathbf{b})$$

$$\frac{a\#t'\ a\#t}{a\#t't}\,(\#\mathbf{app}) \qquad \frac{}{a\#\mathsf{c}}\,(\#\mathsf{c})$$

**Figure 2:** Freshness derivation rules for nominal terms

For example from the rules in Figure 2, $\vdash a\#\lambda b.b$, $\vdash a\#\lambda a.a$, and $a\#X \vdash a\#X(\lambda a.Y)$ are all derivable.

**Remark 2.10** Freshness has an accepted technical denotation as a notion in nominal sets, introduced in [GP01], but it has a broader pedigree and it might be convenient to briefly mention it here. Freshness is a notion of *independence*. Notions of dependence and independence have been studied abstractly before; two different examples are in [Coh65, Chapter VII, Section 2] or [Fin85]. Freshness is also related with specific constructions studied for example by Salibra (consider for example the function $\Delta$ in Definition 3 of [Sal00]).

**Definition 2.11** An **equality** is a pair $t = u$. An **axiom** is a pair $\nabla \vdash t = u$ of a freshness context $\nabla$ and an equality $t = u$. We may write $\emptyset \vdash t = u$ as $\vdash t = u$.

Call a set of axioms $\mathsf{T}$ a **theory**. The theories considered in this paper are:

- CORE: the empty set of axioms.
- ULAM: the axioms from Figure 1. In Section 4 we give a formal sense in which this is a nominal algebra theory of the $\lambda$-calculus (with $\alpha\beta$).

  In Figure 1, note that the $a$ and $b$ are *specific* atoms and the $X$, $Z$ and $Z'$ are *specific* unknowns.
- ULAME: the axioms from Figure 1, plus an extra axiom $a\#Z \vdash \lambda a.(Za) = Z$.

  This is, of course, an $\eta$-equality. In Section 5 we give a formal sense in which this is a nominal algebra theory of the $\lambda$-calculus with $\alpha\beta\eta$.

**Definition 2.12** Define **derivability on equalities** by the rules in Figure 3. In this figure, $a$ and $b$ permutatively range over atoms, $t$, $t'$, $u$ and $u'$ range over nominal terms, $X$ over unknowns, $\nabla$ over freshness contexts, $\pi$ over permutations, and $\sigma$ over substitutions.

Write $\Delta \vdash_{\mathsf{T}} t = u$ when a derivation of $t = u$ exists using these rules such that:

- for each instance of $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$, $\nabla \vdash t = u$ is an axiom from $\mathsf{T}$;
- in the derivations of freshnesses (introduced by instances of $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$ and $(\mathbf{perm})$) the freshness assumptions used are from $\Delta$ only.

We write $\emptyset \vdash_{\mathsf{T}} t = u$ as $\vdash_{\mathsf{T}} t = u$.

**Remark 2.13** We discuss the most interesting rules of Figure 3:

- $(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}})$. This **axiom rule** expresses how we obtain instances of axioms: we instantiate unknowns by terms using substitutions (using the substitution action defined in Definition 2.7) *and also* we rename atoms using permutations (using the permutation action defined in Definition 2.5).

$$\frac{}{t=t}\,(\mathbf{refl}) \qquad \frac{t=u}{u=t}\,(\mathbf{symm}) \qquad \frac{t=u \quad u=v}{t=v}\,(\mathbf{tran}) \qquad \frac{a\#t \quad b\#t}{(a\ b)\cdot t=t}\,(\mathbf{perm})$$

$$\frac{t=u}{\lambda a.t=\lambda a.u}\,(\mathbf{cng}\lambda) \qquad \frac{t'=u' \quad t=u}{t't=u'u}\,(\mathbf{cngapp})$$

$$\frac{\{a\#\sigma(X)\mid a\#X\in\nabla\}}{\pi\cdot(t\sigma)=\pi\cdot(u\sigma)}\,(\mathbf{ax}_{\nabla\vdash\mathbf{t=u}}) \qquad \frac{\begin{array}{c}[a\#X]\\ \vdots\\ t=u\end{array}}{t=u}\,(\mathbf{fr}) \quad (a\notin atoms(t,u))$$

**Figure 3:** Derivation rules for nominal equality

This allows ULAM to have finitely many axioms; Figure 1 does not describe axiom-schemes. For example consider the axiom $(\beta\mathbf{var})=(\vdash(\lambda a.a)X=X)$. This uses an arbitrary, but fixed atom $a$ and an arbitrary, but fixed unknown $X$. We deduce $\vdash_{\mathsf{ULAM}}(\lambda b.b)X=X$ using $(\mathbf{ax}_{(\beta\mathbf{var})})$ taking $\pi=(b\ a)$ and $\sigma$ mapping $X$ to $(b\ a)\cdot X$ (and all other $Y$ to $id\cdot Y$) as follows:

$$\frac{}{(\lambda b.b)(b\ a)\cdot((b\ a)\cdot X)=(b\ a)\cdot((b\ a)\cdot X)}\,(\mathbf{ax}_{\vdash(\lambda\mathbf{a.a})\mathbf{X}=\mathbf{X}})$$

The reader can easily check from Definition 2.5 that $(b\ a)\cdot((b\ a)\cdot X)\equiv id\cdot X$.

We might expect the premises of the instance of the axiom rule to be

$$\{\pi(a)\#\pi\cdot\sigma(X)\mid a\#X\in\nabla\}, \quad \text{rather than} \quad \{a\#\sigma(X)\mid a\#X\in\nabla\}.$$

Both versions are correct, because of a known property of nominal terms which we have called *object-level equivariance*:

$$\Delta\vdash a\#t \quad \text{if and only if} \quad \Delta\vdash\pi(a)\#\pi\cdot t$$

for any $\Delta$, $a$, $t$ and $\pi$. This is characteristic of nominal techniques (e.g. [UPG04, Lemma 2.7], [FG07b, Lemma 20], [GM08b, Appendix A], [GP01, Lemma 4.7]).

- $(\mathbf{fr})$. This introduces a fresh atom into the derivation. Square brackets denote *discharge* of the assumption. We can *always* find a fresh atom no matter how unknowns are instantiated, since our syntax is finite and must mention finitely many atoms.

  $(\mathbf{fr})$ adds no deductive power to CORE but it does in the presence of axioms; a full discussion with an example is in [Mat07, Lemma 2.3.18].

- $(\mathbf{perm})$. This rule expresses $\alpha$-equivalence (see Lemma 2.16 and Theorem 3.10). For instance, $(\mathbf{perm})$ allows us to show the following standard $\alpha$-equivalence property:

$$\frac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#\lambda b.b}\,(\#\lambda\mathbf{b}) \qquad \dfrac{}{b\#\lambda b.b}\,(\#\lambda\mathbf{a})}{\lambda a.a=\lambda b.b}\,(\mathbf{perm}) \quad (\lambda a.a\ \equiv\ (a\ b)\cdot\lambda b.b)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{a\#b}\,(\#\mathbf{ab})}{a\#\lambda c.b}\,(\#\lambda\mathbf{b}) \quad \dfrac{}{c\#\lambda c.b}\,(\#\lambda\mathbf{a})}{\lambda a.b = \lambda c.b}\,(\mathbf{perm})}{\lambda b.\lambda a.b = \lambda b.\lambda c.b}\,(\mathbf{cng}\lambda) \quad \dfrac{}{a=a}\,(\mathbf{refl})}{(\lambda b.(\lambda a.b))a = (\lambda b.(\lambda c.b))a}\,(\mathbf{cngapp}) \quad \dfrac{\dfrac{\dfrac{}{c\#a}\,(\#\mathbf{ab})}{(\lambda b.(\lambda c.b))a = \lambda c.((\lambda b.b)a)}\,(\mathbf{ax}_{\beta\mathbf{abs}}) \quad \dfrac{}{(\lambda b.b)a = a}\,(\mathbf{ax}_{\beta\mathbf{var}})}{\lambda c.((\lambda b.b)a) = \lambda c.a}\,(\mathbf{cng}\lambda)}{(\lambda b.(\lambda c.b))a = \lambda c.a}\,(\mathbf{tran})}{(\lambda b.(\lambda a.b))a = \lambda c.a}\,(\mathbf{tran})$$

**Figure 4:** $\beta$-equality with an $\alpha$-conversion

$(\mathbf{perm})$ captures several rules from [UPG04, Figure 2] (but not in a syntax-directed manner).

- $(\mathbf{refl})$. Choosing $a, b \notin \mathit{atoms}(t)$ we can construct the derivation sketched below:

$$\dfrac{\dfrac{\dfrac{\vdots}{a\#(a\ b)\cdot t} \quad \dfrac{\vdots}{b\#(a\ b)\cdot t}}{t = (a\ b)\cdot t}\,(\mathbf{perm}) \quad \dfrac{\dfrac{\vdots}{a\#t} \quad \dfrac{\vdots}{b\#t}}{(a\ b)\cdot t = t}\,(\mathbf{perm})}{\dfrac{t=t}{t=t}\,(\mathbf{fr})}\,(\mathbf{tran}) \quad \text{(introducing } a\#X, b\#X \text{ for all unknowns } X \text{ in } t)$$

So we can view $(\mathbf{refl})$ as sugar, but (if only for cleaner example derivations) we retain it. All our proofs treat $(\mathbf{refl})$ as a 'real' rule. [8]

**Remark 2.14** Nominal algebra is algebraic (even though the judgement-form $\Delta \vdash t = u$ has '$\Delta \vdash$', which looks like an implication) in the following two senses:

- Nominal algebra is sound and complete for models in nominal sets [GM07,Mat07].
- These models are closed under notions of product, subalgebra, quotient (just like for traditional algebra) — and a nominal sets notion we call *atoms-abstraction*. A version of the HSP theorem (Birkhoff's theorem) holds; any class of nominal algebra models closed under product, subalgebra, quotient, and atoms-abstraction, is characterised by a nominal algebra theory [Gab09].

So, perhaps unexpectedly, nominal algebra retains much of the flavour and mathematical properties of universal algebra.

**Example 2.15** In ULAM we can prove $\beta$-equivalences as illustrated in Figure 4 — we choose one requiring an $\alpha$-conversion.

We now consider some useful examples of derivations in the presence of unknowns:

**Lemma 2.16** $b\#X \vdash_{\mathsf{CORE}} (\lambda a.X)Y = (\lambda b.((b\ a)\cdot X))Y$.

[8] It is a fact that $(\mathbf{symm})$ and $(\mathbf{tran})$ are admissible rules in CORE (a proof can be constructed using Theorem 2.19 below). This property fails in the presence of axioms, for instance, the axioms of ULAM and ULAME. On the other hand, we can view $(\mathbf{refl})$ as sugar for the derivation given above using $(\mathbf{perm})$, in *any* theory.

**Proof** We give the derivation in full:

$$\dfrac{\dfrac{\dfrac{\dfrac{b\#X}{a\#(b\ a)\cdot X}\,(\#\mathbf{X})}{a\#\lambda b.(b\ a)\cdot X}\,(\#\lambda\mathbf{b}) \quad \dfrac{}{b\#\lambda b.(b\ a)\cdot X}\,(\#\lambda\mathbf{a})}{\lambda a.X=\lambda b.(b\ a)\cdot X}\,(\mathbf{perm}) \quad \dfrac{}{Y=Y}\,(\mathbf{refl})}{(\lambda a.X)Y=(\lambda b.(b\ a)\cdot X)Y}\,(\mathbf{cngapp})$$

The instance of $(\mathbf{perm})$ relies on the fact that $(b\ a)\cdot\lambda a.X \equiv \lambda b.(b\ a)\cdot X$. □

Lemma 2.17 is a nominal algebra rendering of the substitution lemma [Bar84, Lemma 2.1.16] in terms of $\beta$-redexes:

**Lemma 2.17** $a\#Y \vdash_{\mathsf{ULAM}} (\lambda b.((\lambda a.Z)X))Y = (\lambda a.((\lambda b.Z)Y))((\lambda b.X)Y)$.

A proof by induction on $Z$ is impossible — $Z$ need not range over syntax, only over elements of nominal algebra models of ULAM (for the general theory of nominal algebra denotations see elsewhere [Mat07]). But ULAM proves this, in logic:

**Proof** By $(\mathbf{tran})$ the proof obligation follows from:

$$(\lambda b.((\lambda a.Z)X))Y = ((\lambda b.(\lambda a.Z))Y)((\lambda b.X)Y) \tag{1}$$

$$((\lambda b.(\lambda a.Z))Y)((\lambda b.X)Y) = (\lambda a.((\lambda b.Z)Y))((\lambda b.X)Y) \tag{2}$$

Part (1) follows by axiom $(\beta\mathbf{app})$; for part (2) we give the full derivation:

$$\dfrac{\dfrac{a\#Y}{(\lambda b.(\lambda a.Z))Y=\lambda a.((\lambda b.Z)Y)}\,(\mathbf{ax}_{\beta\mathbf{abs}}) \quad \dfrac{}{(\lambda b.X)Y=(\lambda b.X)Y}\,(\mathbf{refl})}{((\lambda b.(\lambda a.Z))Y)((\lambda b.X)Y)=(\lambda a.((\lambda b.Z)Y))((\lambda b.X)Y)}\,(\mathbf{cngapp})$$

□

The rules of CORE are not syntax-directed (consider $(\mathbf{tran})$), but we can derive syntactic criteria for equality in CORE, which will also be useful later:

**Definition 2.18** Write $\mathrm{ds}(\pi,\pi')$ for the set $\{a \mid \pi(a)\neq\pi'(a)\}$, the **difference set** of permutations $\pi$ and $\pi'$. We write $\Delta \vdash \mathrm{ds}(\pi,\pi')\#X$ for a set of proof-obligations $\Delta\vdash a\#X$, one for each $a\in\mathrm{ds}(\pi,\pi')$.

**Theorem 2.19** $\Delta\vdash_{\mathsf{CORE}} t=u$ *precisely when one of the following holds:*

- $t\equiv a$ *and* $u\equiv a$.
- $t\equiv\pi\cdot X$, $u\equiv\pi'\cdot X$ *and* $\Delta\vdash\mathrm{ds}(\pi,\pi')\#X$.
- $t\equiv\lambda a.t'$, $u\equiv\lambda a.u'$ *and* $\Delta\vdash_{\mathsf{CORE}} t'=u'$.
- $t\equiv\lambda a.t'$, $u\equiv\lambda b.u'$, $\Delta\vdash b\#t'$ *and* $\Delta\vdash_{\mathsf{CORE}}(b\ a)\cdot t'=u'$.
- $t\equiv t''t'$, $u\equiv u''u'$, $\Delta\vdash_{\mathsf{CORE}} t''=u''$ *and* $\Delta\vdash_{\mathsf{CORE}} t'=u'$.
- $t\equiv\mathsf{c}$ *and* $u\equiv\mathsf{c}$.

For a proof see Appendix A; expanded details are also in [Mat07, Cor. 2.5.4]. Thus, CORE induces the same theory of equality as the rules for equality from [UPG04].

## 3 The $\lambda$-calculus

We give a short formal treatment of $\lambda$-terms and $\alpha\beta(\eta)$-equivalence.

**Definition 3.1** Call a nominal term **ground** when it mentions no unknowns. [9]

As discussed in Subsection 2.1 our nominal terms syntax is specialised to the $\lambda$-calculus; ground terms $g, h, k$ are characterised by:

$$g \;::=\; a \mid \lambda a.g \mid gg \mid \mathsf{c}.$$

Thus, 'ordinary' $\lambda$-terms are a subset of the nominal terms from Definition 2.3, obtained by exluding unknowns. This should be no surprise, since unknowns are in Definition 2.3 to represent meta-variables.

**Definition 3.2** Define the **free atoms** $fa(g)$ by:

$$fa(a) = \{a\} \qquad fa(\lambda a.g) = fa(g) \setminus \{a\} \qquad fa(g'g) = fa(g') \cup fa(g) \qquad fa(\mathsf{c}) = \emptyset$$

**Lemma 3.3** *$a \notin fa(g)$ if and only if $\vdash a\#g$.*
*Also, if $a \notin atoms(g)$ then $\vdash a\#g$.*

**Definition 3.4** Define the **size** $|g|$ of a ground term $g$ by:

$$|a| = 1 \qquad |\lambda a.g| = |g| + 1 \qquad |g'g| = |g'| + |g| + 1 \qquad |\mathsf{c}| = 1$$

**Definition 3.5** Fix an infinite set of atoms $\mathcal{F} \subseteq \mathbb{A}$, such that $\mathbb{A} \setminus \mathcal{F}$ is also infinite.

**Definition 3.6** We define a **capture-avoiding substitution** action $g[h/a]$ inductively on $|g|$ by:

$$\begin{aligned}
a[h/a] &\equiv h \\
b[h/a] &\equiv b \\
(\lambda a.g)[h/a] &\equiv \lambda a.g \\
(\lambda b.g)[h/a] &\equiv \lambda b.(g[h/a]) && (b \notin fa(h)) \\
(\lambda b.g)[h/a] &\equiv \lambda b'.(g[b'/b][h/a]) && (b \in fa(h),\ b' \text{ fresh, in } \mathcal{F}) \\
(g'g)[h/a] &\equiv (g'[h/a])(g[h/a]) \\
\mathsf{c}[h/a] &= \mathsf{c}
\end{aligned}$$

In the clause for $(\lambda b.g)[h/a]$ we make some fixed but arbitrary choice of fresh $b' \in \mathcal{F}$ for each $b, g, h, a$ (so $b' \in \mathcal{F} \setminus atoms(g) \cup atoms(h) \cup \{a, b\}$).

**Remark 3.7** We fix $\mathcal{F}$ and make our choice of fresh $b'$ be *from* $\mathcal{F}$ because this is convenient for our later proof of completeness of ULAM. This choice allows us to prove Lemma 3.15, which is needed to prove Lemma 4.13, and this in turn guarantees Lemma 4.17, which makes it easier for us to express a compact definition of the inverse translation in Definition 4.18, which is a key component of the proof of Completeness (Theorem 4.7). This pattern is repeated in Section 5.

[9] Ground terms should not be confused with closed lambda terms, i.e. terms without free atoms; closed terms are not used in this paper.

**Definition 3.8** Let **$\alpha$-equivalence** $=_\alpha$ be the least transitive reflexive symmetric relation such that:

- (Name-abstraction.)
  If $g[c/a] =_\alpha g'[c/b]$ for fresh $c$ (so $c \notin atoms(g, g')$) then $\lambda a.g =_\alpha \lambda b.g'$.
- (Congruence for $\lambda$.) If $g =_\alpha g'$ then $\lambda a.g =_\alpha \lambda a.g'$.
- (Congruence for application.) If $g =_\alpha g'$ and $h =_\alpha h'$ then $gh =_\alpha g'h'$.

**Lemma 3.9** *Suppose $g$ is a ground term.*

(i) *If $a, b \notin fa(g)$ then $(a\ b) \cdot g =_\alpha g$.*

(ii) *If $b \notin fa(g)$ then $g[b/a] =_\alpha (b\ a) \cdot g$.*

**Proof** For the first part, we observe that since $a, b \notin fa(g)$, any $a$ and $b$ that occur in $g$ must occur in the scope of $\lambda a$ and $\lambda b$. We traverse the structure of $g$ bottom-up and rename these to fresh atoms (for example $\lambda a'$ and $\lambda b'$ which do not occur anywhere in $g$). Call the resulting term $g'$. Now $(a\ b) \cdot g' \equiv g'$ because $a, b \notin atoms(g')$. Equality is symmetric, so we reverse the process to return to $g$.

The second part then follows by an induction on $|g|$. □

**Theorem 3.10** *On ground terms, derivable equality in* CORE *coincides with $=_\alpha$.*

(See also [Mat07, Theorem 4.3.13].)

**Proof** We must show that for ground terms $g, h$,

$$\vdash_{\mathsf{CORE}} g = h \quad \text{if and only if} \quad g =_\alpha h.$$

We prove the left-to-right implication by induction on the structure of $g$, using the syntactic criteria for CORE-equality (Theorem 2.19). The cases of $g \equiv a$ and $g \equiv \mathsf{c}$ follow by reflexivity, and the case of $g \equiv g''g'$ follows by congruence using the inductive hypothesis. Now suppose $g \equiv \lambda a.g'$, then there are two possibilities:

(i) $h \equiv \lambda a.h'$ and $\vdash_{\mathsf{CORE}} g' = h'$. Then $g' =_\alpha h'$ by the inductive hypothesis, and we conclude $\lambda a.g' =_\alpha \lambda a.h'$ by congruence.

(ii) $h \equiv \lambda b.h'$, $\vdash b\#g'$ and $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$. By Lemma 3.3 and some easy calculations we know $a, b \notin fa(\lambda a.g')$, so $\lambda a.g' =_\alpha \lambda b.(b\ a) \cdot g'$ by part 1 of Lemma 3.9 and symmetry. Also $\lambda b.(b\ a) \cdot g' =_\alpha \lambda b.h'$ by congruence and the inductive hypothesis. We conclude $\lambda a.g' =_\alpha \lambda b.h'$ by transitivity.

Conversely suppose that $g =_\alpha h$. It suffices to show that equality in CORE can simulate every derivation rule of $=_\alpha$. We treat the only non-trivial case.

Suppose we have deduced $\lambda a.g =_\alpha \lambda b.h$ from $g[c/a] =_\alpha h[c/b]$, where $c$ is fresh (so $c \notin atoms(g, h)$). By Lemma 3.3 we know $\vdash c\#g$ and $\vdash c\#h$, with which we can show

$$\vdash_{\mathsf{CORE}} g[c/a] = (c\ a) \cdot g \qquad \text{and} \qquad \vdash_{\mathsf{CORE}} h[c/b] = (c\ b) \cdot h$$

by an induction on $|g|$ and $|h|$.

Now also $\vdash_{\mathsf{CORE}} g[c/a] = h[c/b]$ by the inductive hypothesis. Then we obtain

$$\vdash_{\mathsf{CORE}} \lambda c.(c\ a) \cdot g = \lambda c.(c\ b) \cdot h.$$

by $(\mathbf{symm})$, $(\mathbf{tran})$ and $(\mathbf{cong}[])$.

By $(\mathbf{perm})$ $\vdash_{\mathsf{CORE}} \lambda c.(c\ a)\cdot g = \lambda a.g$ and $\vdash_{\mathsf{CORE}} \lambda c.(c\ b)\cdot h = \lambda b.h$, since $\vdash c\#g$ and $\vdash c\#h$. Using $(\mathbf{symm})$ and $(\mathbf{tran})$ we conclude that $\vdash_{\mathsf{CORE}} \lambda a.g = \lambda b.h$. □

**Remark 3.11** We do not quotient terms by $\alpha$-conversion and we do not use a nominal-style datatype of syntax-with-binding [GP01]. Later on the proof-method for Theorem 4.7 involves delicate accounting of what atoms appear abstracted in terms. In particular, we do not want to have to invent names (and keep track of our invented names) for abstracted atoms in Definition 4.18.

We conclude this section with some basic definitions and lemmas, which will be useful later.

**Definition 3.12** Let (one step) $\beta$**-reduction** $g \to_\beta h$ be defined by:

- $(\lambda a.g)h \to_\beta g[h/a]$.
- If $g \to_\beta g'$ then $\lambda a.g \to_\beta \lambda a.g'$.
- If $g \to_\beta g'$ then $gh \to_\beta g'h$.
- If $h \to_\beta h'$ then $gh \to_\beta gh'$.

Let (one step) $\beta$**-equality** $\leftrightarrow_\beta$ be defined by $g \leftrightarrow_\beta h$ when $g \to_\beta h$ or $h \to_\beta g$.

Let (multi step) $\alpha\beta$**-equality** $=_{\alpha\beta}$ be the least transitive reflexive relation containing $\leftrightarrow_\beta$ and $=_\alpha$.

**Definition 3.13** Let (one step) $\eta$**-contraction** be defined by:

- $\lambda a.(ga) \to_\eta g$ if $a \notin fa(g)$.
- If $g \to_\eta g'$ then $\lambda a.g \to_\eta \lambda a.g'$.
- If $g \to_\eta g'$ then $gh \to_\eta g'h$.
- If $h \to_\eta h'$ then $gh \to_\eta gh'$.

Let (one step) $\eta$**-equality** be defined by $g \leftrightarrow_\eta h$ when $g \to_\eta h$ or $h \to_\eta g$.

Let (multi step) $\alpha\beta\eta$**-equality** $=_{\alpha\beta\eta}$ be the least transitive reflexive relation containing $\leftrightarrow_\eta$, $\leftrightarrow_\beta$, and $=_\alpha$.

**Definition 3.14** Define the **bound atoms** $ba(g)$ by:

$$ba(a) = \emptyset \qquad ba(\lambda a.g) = ba(g) \cup \{a\} \qquad ba(g'g) = ba(g') \cup ba(g) \qquad ba(\mathsf{c}) = \emptyset$$

**Lemma 3.15** *For ground terms $g$, $g'$:*

(i) *If $g \to_\beta g'$ and $ba(g) \subseteq \mathcal{F}$, then $ba(g') \subseteq \mathcal{F}$.*

(ii) *If $g \to_\eta g'$ and $ba(g) \subseteq \mathcal{F}$, then $ba(g') \subseteq \mathcal{F}$.*

**Proof** The first part is by routine calculations, using the fact that in Definition 3.6 we choose fresh atoms from $\mathcal{F}$.

The second part is trivial, since $\eta$-contraction eliminates a bound variable. □

**Lemma 3.16** *For ground terms $g$, $g'$, $h$:*

(i) *If $g =_\alpha g'$ and $g \to_\beta h$ then there exists some $h'$ such that $g' \to_\beta h'$ and $h' =_\alpha h$.*

(ii) *If $g =_\alpha g'$ and $g \to_\eta h$ then there exists some $h'$ such that $g' \to_\eta h'$ and $h' =_\alpha h$.*

# 4 Soundness, completeness, and conservativity for $\alpha\beta$

In Section 2 we presented nominal algebra and the theory ULAM. We saw formal derivations reminiscent of the 'informal meta-level'. This informal meta-level is made formal using nominal terms; object-level variables become atoms, and meta-level variables become unknowns. ULAM axiomatises the $\lambda$-calculus up to $\alpha\beta$-equivalence within this framework. Theorems 4.3 and 4.7 make that formal.

## *4.1 Soundness*

**Definition 4.1** Call $\sigma$ a **ground substitution** for a set of unknowns $\mathcal{X}$ when $\sigma(X)$ is ground for every $X \in \mathcal{X}$. Call $\sigma$ ground for $\Delta, t, u$ when $\sigma$ is ground for the set of unknowns appearing anywhere in $\Delta$, $t$, or $u$.

So: ground substitutions eliminate all metavariables.

**Definition 4.2** Write $\Delta \models_{\alpha\beta} t = u$ when $t\sigma =_{\alpha\beta} u\sigma$ (Definition 3.12) for all ground substitutions $\sigma$ for $\Delta, t, u$ such that $a \notin fa(\sigma(X))$ for every $a\#X \in \Delta$.

**Theorem 4.3 (Soundness)** *For any $\Delta$, $t$, $u$, if $\Delta \vdash_{\mathsf{ULAM}} t = u$ then $\Delta \models_{\alpha\beta} t = u$.*

**Proof** We proceed by induction on ULAM derivations. We sketch the proof (some reasoning on freshnesses is elided):

- The cases $(\mathbf{refl})$, $(\mathbf{symm})$, $(\mathbf{tran})$, $(\mathbf{cng}\lambda)$ and $(\mathbf{cngapp})$ follow by induction using the fact that $=_{\alpha\beta}$ is an equivalence relation and a congruence.
- The case $(\mathbf{perm})$. Suppose $a, b \notin fa(g)$. By part 1 of Lemma 3.9, $(a\ b) \cdot g =_{\alpha} g$ and $(a\ b) \cdot g =_{\alpha\beta} g$ follows.
- The case $(\mathbf{fr})$. Unknowns are irrelevant because ground terms by definition do not contain them. If $\sigma(X)$ mentions an atom which $(\mathbf{fr})$ generates fresh for some $X$ in $\Delta$, $t$, or $u$, then we 'freshen' the atom further to avoid an 'name clash'. [10]
- The case $(\mathbf{ax})$. The axioms of ULAM are all standard properties of the $\lambda$-calculus:
  - $(\lambda a.a)h =_{\alpha\beta} h$.
  - If $a \notin fa(g)$ then $(\lambda a.g)h =_{\alpha\beta} g$.
  - $(\lambda a.(g'g))h =_{\alpha\beta} (\lambda a.g')h)((\lambda a.g)h)$.
  - If $b \notin fa(h)$ then $(\lambda a.(\lambda b.g))h =_{\alpha\beta} \lambda b.((\lambda a.g)h)$.
  - $(\lambda a.g)a =_{\alpha\beta} g$.

□

## *4.2 Completeness and conservativity*

Recall our choice of $\mathcal{F}$ from Definition 3.5:

**Definition 4.4** Fix a freshness context $\Delta$ and two terms $t$ and $u$. Let $\mathcal{A}$ be the atoms mentioned anywhere in $\Delta$, $t$, or $u$, i.e. $\mathcal{A} = \{a \mid a\#X \in \Delta\} \cup atoms(t, u)$. Let $\mathcal{X}$ be the unknowns mentioned anywhere in $\Delta$, $t$, or $u$. For each $X \in \mathcal{X}$ fix the following data:

[10] We retain the inductive hypothesis of the 'freshened' derivation using the mathematical principle of ZFA equivariance [GM08b, Appendix A] — or by performing induction instead on the depth of derivations, and proving that freshening atoms does not affect this measure.

- an order $a_{X1}, \ldots, a_{Xk_X}$ on the atoms in $\mathcal{A}$ such that $a\#X \notin \Delta$;
- two entirely fresh atoms $d_X$ and $e_X$ (so $d_X \notin \mathcal{F} \cup \mathcal{A}$, and $e_X \notin \mathcal{F} \cup \mathcal{A}$).

Write $\mathcal{D}$ for $\{d_X \mid X \in \mathcal{X}\}$ and $\mathcal{E}$ for $\{e_X \mid X \in \mathcal{X}\}$.

**Definition 4.5** Specify $\varsigma$ a ground substitution for $\mathcal{X}$ by:

- $\varsigma(X) \equiv e_X(d_X a_{X1} \ldots a_{Xk_X})$ when $X \in \mathcal{X}$, and
- $\varsigma(X) \equiv X$ otherwise (the choice of $X$ in the right-hand side is irrelevant).

Lemma 4.6 is easy and will be useful:

**Lemma 4.6** *If $a\#X \in \Delta$ then $a \notin fa(\varsigma(X))$.*

**Proof** By construction of $\Delta$ and $\varsigma$ (Definitions 4.4 and 4.5) $a$ differs from all of the $a_{Xi}$. □

**Theorem 4.7 (Completeness)** *If $\Delta \models_{\alpha\beta} t = u$ then $\Delta \vdash_{\mathsf{ULAM}} t = u$.*

The proof of this result uses the $\varsigma$ constructed above, and it is technical. Therefore, we defer it to Subsection 4.3 and mention Corollary 4.8 and Theorem 4.12, which are two forms of conservativity result for ULAM over CORE.

**Corollary 4.8** *Suppose $g$ and $h$ are ground. Then $\vdash_{\mathsf{ULAM}} g = h$ if and only if $g =_{\alpha\beta} h$.*

**Proof** By Theorems 4.3 and 4.7, using that fact that $g\varsigma \equiv g$ and $h\varsigma \equiv h$. □

**Definition 4.9** Call a ground term $g$ a **$\beta$-normal form** when no $g'$ exists with $g \to_\beta g'$.

**Lemma 4.10** *Fix $\Delta$. Suppose that $t$ and $u$ contain no subterm of the form $(\lambda a.v)w$. Then for $\varsigma$ the ground substitution from Definition 4.5, $t\varsigma$ and $u\varsigma$ are $\beta$-normal forms.*

**Proof** $\varsigma(X) \equiv e_X(d_X a_{X1} \ldots a_{Xk_X})$ for every $X$ appearing in $\Delta$, $t$, $u$. Applying this substitution to $t$ and $u$ cannot introduce subterms of the form $(\lambda a.v)w$. □

**Lemma 4.11** *$t\varsigma =_\alpha u\varsigma$ implies $\Delta \vdash_{\mathsf{CORE}} t = u$.*

**Proof** We prove by induction on $t'$ that if $t'$ is a subterm of $t$ and $u'$ a subterm of $u$ then $t'\varsigma =_\alpha u'\varsigma$ implies $\Delta \vdash_{\mathsf{CORE}} t' = u'$.

The interesting case is when $t' \equiv \pi \cdot X$. Suppose $e_X(d_X\pi(a_{X1}) \ldots \pi(a_{Xk_X}) =_\alpha u'\varsigma)$. Then it *must* be that $u'\varsigma \equiv e_X(d_X\pi(a_{X1}) \ldots \pi(a_{Xk_X}))$. By the construction of $u'\varsigma$ and the way we chose $a_{X1}, \ldots, a_{Xk_X}$ to be the atoms mentioned in $\Delta$, $t$, or $u$ which are *not* provably fresh for $X$ in $\Delta$, it follows that $u'$ must take the form $\pi' \cdot X$, for some $\pi'$ such that $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$. It follows that $\Delta \vdash_{\mathsf{CORE}} t' = u'$ as required. □

**Theorem 4.12 (Conservativity)** *Suppose that $t$ and $u$ contain no subterm of the form $(\lambda a.v)w$. Then*

$$\Delta \vdash_{\mathsf{ULAM}} t = u \qquad \textit{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

**Proof** A derivation in CORE is also a derivation in ULAM so the right-to-left implication is immediate.

Now suppose that $\Delta \vdash_{\mathsf{ULAM}} t = u$. We construct $\varsigma$ as in Definition 4.5. By Theorem 4.3, $t\varsigma =_{\alpha\beta} u\varsigma$. By Lemma 4.10 we know that $t\varsigma$ and $u\varsigma$ are $\beta$-normal forms. By confluence [Bar84, Theorem 3.2.8 (i)] $t\varsigma =_\alpha u\varsigma$. By Lemma 4.11 $\Delta \vdash_{\mathsf{CORE}} t = u$, as required. □

### 4.3 *Proof of Theorem 4.7*

Recall the definition of $ba(g)$ from Definition 3.14.

**Lemma 4.13** *There exists a chain*

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma$$

*where each $\leftrightarrow_?$ is one of $=_\alpha$ or $\leftrightarrow_\beta$, which is such that $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for $1 \leq i \leq m$ (so none of the terms in the chain contains '$\lambda d_X$' or '$\lambda e_X$' for any $d_X \in \mathcal{D}$ or $e_X \in \mathcal{E}$).*

**Proof** By assumption $\Delta \models_{\alpha\beta} t = u$, so by Lemma 4.6, we know $t\varsigma =_{\alpha\beta} u\varsigma$. It follows that there is a chain

$$t\varsigma \equiv g'_1 \leftrightarrow_? g'_2 \leftrightarrow_? g'_3 \leftrightarrow_? \ldots \leftrightarrow_? g'_{m'-1} \leftrightarrow_? g'_{m'} \equiv u\varsigma$$

where each $\leftrightarrow_?$ is one of $=_\alpha$ or $\leftrightarrow_\beta$.

By construction, $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for $i = 1$ and $i = m'$. We transform this into a chain such that all terms satisfy this property.

Suppose $h \rightarrow_\beta h'$ is a link in the chain and such that $ba(h) \cap (\mathcal{D} \cup \mathcal{E}) \neq \emptyset$ (so $h \equiv g'_i$ and $h' \equiv g'_{i+1}$, or $h \equiv g'_i$ and $h' \equiv g'_{i-1}$, for some $i$). We use Lemma 3.16 to replace this link with $h =_\alpha h'' \rightarrow_\beta h''' =_\alpha h'$, where we choose $h''$ such that $ba(h'') \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$. By part 1 of Lemma 3.15 also $ba(h''') \cap (\mathcal{D} \cup \mathcal{E}) \neq \emptyset$.

We iterate the replacement above as much as possible. Now suppose $h =_\alpha h' =_\alpha h''$ is a pair of links in the chain. Since $=_\alpha$ is transitive, we replace this with $h =_\alpha h''$.

It is easy to check that the final chain has the form

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma$$

where $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for all $1 \leq i \leq m$, as required. □

We can make the following definition:

**Definition 4.14** *Make a fixed but arbitrary choice of chain as specified in Lemma 4.13.*

**Definition 4.15** Let $\mathcal{A}^+$ be the set of all atoms mentioned anywhere in the chain we fixed in Definition 4.14, and let $\Delta^+$ be $\Delta$ enriched with freshness assumptions $a\#X$ for every $a \in \mathcal{A}^+ \setminus \mathcal{A}$ and every $X \in \mathcal{X}$.

**Definition 4.16** Call a ground term $g$ **accurate** when:

- $atoms(g) \subseteq \mathcal{A}^+$ ($g$ mentions only atoms in $\mathcal{A}^+$); and
- $ba(g) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ ($g$ does not contain '$\lambda d_X$' or '$\lambda e_X$' for any $d_X \in \mathcal{D}$ or $e_X \in \mathcal{E}$).

**Lemma 4.17** *$g_1, \ldots, g_m$ are accurate.*

**Proof** Trivially, from the construction in Lemma 4.13 and from our choice of $\mathcal{A}^+$. □

**Definition 4.18** Define an **inverse translation** from accurate ground terms to (possibly non-ground) terms inductively by:

$$a^{-1} \equiv a \quad (a \notin \mathcal{D} \cup \mathcal{E}) \qquad (\lambda a.g)^{-1} \equiv \lambda a.(g^{-1}) \qquad (gh)^{-1} \equiv (g^{-1})(h^{-1}) \qquad \mathsf{c}^{-1} \equiv \mathsf{c}$$

$$(d_X)^{-1} \equiv \lambda a_{X1}.\cdots\lambda a_{Xk_X}.X \ \ (d_X \in \mathcal{D}) \qquad (e_X)^{-1} \equiv \lambda e_X.e_X \ \ (e_X \in \mathcal{E})$$

We need some technical lemmas:

**Lemma 4.19** *Suppose that $g$ is accurate. For any $a \in \mathcal{A}^+$, if $a \notin fa(g)$ then $\Delta^+ \vdash a\#g^{-1}$.*

**Proof** By induction on $g$. The non-trivial case is when $g \equiv d_X$. If $a \notin fa(d_X)$ then $a \neq d_X$ and we must show

$$\Delta^+ \vdash a\#\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X,$$

which follows using the rules for freshness and the fact that the atoms that might not be fresh for $X$ are precisely the $a_{Xi}$. □

**Lemma 4.20** *Suppose that $g$ is accurate. Suppose that $\pi$ is a permutation such that $\pi(a) = a$ for all $a \notin \mathcal{A}^+ \setminus (\mathcal{D} \cup \mathcal{E})$. Then $\Delta^+ \vdash_{\mathsf{CORE}} (\pi \cdot g)^{-1} = \pi \cdot (g^{-1})$.*

**Proof** By a routine induction on $g$. In the case of $g \equiv d_X$ we use the fact that $\pi(a) = a$ for all $a \in (\mathcal{D} \cup \mathcal{E})$. □

**Lemma 4.21** *Suppose that $g$ and $h$ are accurate. Then $g =_\alpha h$ implies $\Delta^+ \vdash_{\mathsf{CORE}} g^{-1} = h^{-1}$.*

**Proof** By Theorem 3.10 $g =_\alpha h$ coincides with $\vdash_{\mathsf{CORE}} g = h$. The proof is then by a detailed but routine induction on $g$ using the syntactic criteria of Theorem 2.19.

The only non-trivial case is when

$$g \equiv \lambda a.g', \quad h \equiv \lambda b.h', \quad \vdash b\#g', \quad \text{and} \quad \vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'.$$

We must show $\Delta^+ \vdash_{\mathsf{CORE}} \lambda a.(g')^{-1} = \lambda b.(h')^{-1}$. By transitivity and congruence, it suffices to show the following:

- $\Delta^+ \vdash_{\mathsf{CORE}} \lambda a.(g')^{-1} = \lambda b.(b\ a) \cdot (g')^{-1}$. By $(\mathbf{perm})$ and the rules for freshness, this follows from $\Delta^+ \vdash b\#(g')^{-1}$. Using Lemma 4.19, this follows from our assumption that $\vdash b\#g'$.
- $\Delta^+ \vdash_{\mathsf{CORE}} (b\ a) \cdot (g')^{-1} = ((b\ a) \cdot g')^{-1}$. $\lambda a.g'$ and $\lambda b.h'$ are accurate so $a, b \in \mathcal{A}^+ \setminus (\mathcal{D} \cup \mathcal{E})$. The result follows by Lemma 4.20.
- $\Delta^+ \vdash_{\mathsf{CORE}} ((b\ a) \cdot g')^{-1} = (h')^{-1}$. $(b\ a) \cdot g'$ and $h'$ are accurate, so this follows by inductive hypothesis from our assumption that $\vdash_{\mathsf{CORE}} (b\ a) \cdot g' = h'$.

□

**Lemma 4.22** *Suppose that $a \in \mathcal{A}^+ \setminus (\mathcal{D} \cup \mathcal{E})$. Suppose that $g$, $h$, and $g[h/a]$ are accurate. Then $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(g^{-1}))(h^{-1}) = (g[h/a])^{-1}$.*

**Proof** By induction on $|g|$ (Definition 3.4). We consider the cases in turn:

- $a[h/a]$. $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.a)(h^{-1}) = h^{-1}$ by axiom $(\beta\mathbf{var})$.
- $b[h/a]$ where $b \notin (\mathcal{D} \cup \mathcal{E})$. So $\Delta^+ \vdash a\#b$, and $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.b)(h^{-1}) = b$ by axiom $(\beta\#)$. (Recall that by our permutative convention, $b$ ranges over atoms other than $a$.)
- $\mathsf{c}[h/a]$. So $\Delta^+ \vdash a\#\mathsf{c}$, and $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.\mathsf{c})(h^{-1}) = \mathsf{c}$ by axiom $(\beta\#)$. (Recall from Definition 2.1 that $\mathsf{c}$ ranges over constant symbols.)
- $d_X[h/a]$ where $d_X \in \mathcal{D}$. By assumption $a \neq d_X$, so we must show

$$\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X))(h^{-1}) = \lambda a_{X1}.\cdots\lambda a_{Xk_X}.X.$$

Using axiom $(\beta\#)$ (and $(\mathbf{tran})$ and $(\mathbf{refl})$) it suffices to show

$$\Delta^+ \vdash a\#\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X.$$

This follows from Lemma 4.19 and the fact that $a \notin fa(d_X) = \{d_X\}$.

- $e_X[h/a]$ where $e_X \in \mathcal{E}$. By assumption $a \neq e_X$, so we must show

$$\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(\lambda e_X.e_X))(h^{-1}) = \lambda e_X.e_X.$$

Using axiom $(\beta\#)$ (and $(\mathbf{tran})$ and $(\mathbf{refl})$) it suffices to show

$$\Delta^+ \vdash a\#\lambda e_X.e_X.$$

We derive this using the derivation rules for freshness (Figure 2).

- $(\lambda a.g)[h/a]$. $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(\lambda a.(g)^{-1}))(h^{-1}) = \lambda a.(g^{-1})$ by axiom $(\beta\#)$.
- $(\lambda b.g)[h/a]$ where $b \notin fa(h)$. $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(\lambda b.(g)^{-1}))(h^{-1}) = \lambda b.((g[h/a])^{-1})$ by the following ULAM derivation, presented in a calculational style:

$$\begin{aligned}
&(\lambda a.(\lambda b.(g)^{-1}))(h^{-1})\\
=\ &\{\text{ axiom } (\beta\mathbf{abs}), \text{ since } \Delta^+ \vdash b\#h^{-1} \text{ by Lemma 4.19 }\}\\
&\lambda b.((\lambda a.(g)^{-1})(h^{-1}))\\
=\ &\{\text{ inductive hypothesis }\}\\
&\lambda b.((g[h/a])^{-1})
\end{aligned}$$

- $(\lambda b.g)[h/a]$ where $b \in fa(h)$. By assumption $\lambda b.g$ is accurate, therefore $b \notin (\mathcal{D} \cup \mathcal{E})$. By Definition 3.6, $(\lambda b.g)[h/a] \equiv \lambda b'.(g[b'/b][h/a])$ for fresh $b'$ (so $b' \notin atoms(g,h)$). By assumption $\lambda b'.(g[b'/b][h/a])$ is accurate, and so $b' \notin (\mathcal{D} \cup \mathcal{E})$ and $g[b'/b][h/a]$ is accurate. We must show

$$\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(\lambda b.(g^{-1})))(h^{-1}) = \lambda b'.((g[b'/b][h/a])^{-1}).$$

Note that by Lemma 4.19, $\Delta^+ \vdash b'\#g^{-1}$ and $\Delta^+ \vdash b'\#h^{-1}$, and $\Delta^+ \vdash b'\#\lambda b.(g^{-1})$ follows by $(\#\lambda\mathbf{b})$. Also $\Delta^+ \vdash b\#\lambda b.(g^{-1})$ is immediate by $(\#\lambda\mathbf{a})$. We present the rest of the proof as a calculation:

$$\begin{aligned}
&\lambda b'.((g[b'/b][h/a])^{-1})\\
=\ &\{\ g[b'/b] =_\alpha (b'\ b)\cdot g \text{ by part 2 of Lemma 3.9 since } b' \notin fa(g)\ \}\\
&\lambda b'.(((b'\ b)\cdot g)[h/a])^{-1}\\
=\ &\{\text{ inductive hypothesis, since } (b'\ b)\cdot g \text{ is accurate }\}\\
&\lambda b'.((\lambda a.((b'\ b)\cdot g)^{-1})(h^{-1}))\\
=\ &\{\text{ Lemma 4.20 }\}\\
&\lambda b'.((\lambda a.(b'\ b)\cdot(g^{-1}))(h^{-1}))\\
=\ &\{\text{ axiom } (\beta\mathbf{abs}), \text{ since } \Delta^+ \vdash b'\#h^{-1}\ \}\\
&(\lambda a.(\lambda b'.(b'\ b)\cdot(g^{-1})))(h^{-1})\\
=\ &\{\ (\mathbf{perm}) \text{ since } \Delta^+ \vdash b\#\lambda b.(g^{-1}) \text{ and } \Delta^+ \vdash b'\#\lambda b.(g^{-1})\ \}\\
&(\lambda a.(\lambda b.(g^{-1})))(h^{-1})
\end{aligned}$$

- $(g'g)[h/a]$. By axiom $(\beta\mathbf{app})$ and the inductive hypothesis.

The result follows. □

**Corollary 4.23** *Suppose that $g$ and $h$ are accurate. If $g \leftrightarrow_\beta h$ then $\Delta^+ \vdash_{\mathsf{ULAM}} g^{-1} = h^{-1}$.*

**Proof** By induction on the rules for $\to_\beta$ from Definition 3.12. It suffices to show the following (here $g$, $g'$, $h$, $h'$, and $g[h/a]$ are accurate and $a \in \mathcal{A}^+ \setminus (\mathcal{D} \cup \mathcal{E})$):

- $\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda a.(g^{-1}))(h^{-1}) = (g[h/a])^{-1}$.
- If $\Delta^+ \vdash_{\mathsf{ULAM}} g^{-1} = (g')^{-1}$ then $\Delta^+ \vdash_{\mathsf{ULAM}} \lambda a.(g^{-1}) = \lambda a.((g')^{-1})$.
- If $\Delta^+ \vdash_{\mathsf{ULAM}} g^{-1} = (g')^{-1}$ then $\Delta^+ \vdash_{\mathsf{ULAM}} (g^{-1})(h^{-1}) = ((g')^{-1})(h^{-1})$.
- If $\Delta^+ \vdash_{\mathsf{ULAM}} h^{-1} = (h')^{-1}$ then $\Delta^+ \vdash_{\mathsf{ULAM}} (g^{-1})(h^{-1}) = (g^{-1})((h')^{-1})$.

The first part is Lemma 4.22. The other parts follow by $(\mathbf{cng}\lambda)$ and $(\mathbf{cngapp})$. □

**Lemma 4.24** *$\Delta^+ \vdash_{\mathsf{ULAM}} (t\varsigma)^{-1} = t$, and $\Delta^+ \vdash_{\mathsf{ULAM}} (u\varsigma)^{-1} = u$.*

**Proof** We prove by induction that if $v$ is a subterm of $t$ or $u$ then $\Delta^+ \vdash_{\mathsf{ULAM}} (v\varsigma)^{-1} = v$ is derivable.

The only interesting case is when $v \equiv \pi \cdot X$. We must show that

$$\Delta^+ \vdash_{\mathsf{ULAM}} (\lambda e_X.e_X)\big((\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X)\pi(a_{X1})\cdots\pi(a_{Xk_X})\big) = \pi \cdot X.$$

Take a set of fresh atoms $\mathcal{B} = \{b_{Xi} \mid X, i \text{ such that } a_{Xi} \in \mathcal{A}\}$ in bijection with $\mathcal{A}$; so $\mathcal{B}$ is disjoint from the $a_{Xi}$ and $\pi(a_{Xi})$. Let $\Delta_\mathcal{B} = \{b_{Xi}\#Y \mid b_{Xi} \in \mathcal{B} \text{ and } Y \in \mathcal{X}\}$. Then by $(\mathbf{fr})$, it suffices to show

$$\Delta^+ \cup \Delta_\mathcal{B} \vdash_{\mathsf{ULAM}} (\lambda e_X.e_X)\big((\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X)\pi(a_{X1})\cdots\pi(a_{Xk_X})\big) = \pi \cdot X.$$

Now let $\pi_1 = (b_{X1}\ a_{X1})\cdots(b_{Xk_X}\ a_{Xk_X})$ and $\pi_2 = (b_{Xk_X}\ \pi(a_{Xk_X}))\cdots(b_{X1}\ \pi(a_{X1}))$. We will need the following properties of derivability in CORE involving these permutations:

$$\Delta^+ \cup \Delta_\mathcal{B} \ \vdash_{\mathsf{CORE}} \ \lambda a_{X1}.\cdots\lambda a_{Xk_X}.X = \lambda b_{X1}.\cdots\lambda b_{Xk_X}.\pi_1 \cdot X \tag{3}$$

$$\Delta^+ \cup \Delta_\mathcal{B} \ \vdash_{\mathsf{CORE}} \ \lambda b_{X1}.\cdots\lambda b_{Xk_X}.\pi_1 \cdot X = \lambda\pi(a_{X1}).\cdots\lambda\pi(a_{Xk_X}).(\pi_2 \circ \pi_1) \cdot X \tag{4}$$

$$\Delta^+ \cup \Delta_\mathcal{B} \ \vdash_{\mathsf{CORE}} \ (\pi_2 \circ \pi_1) \cdot X = \pi \cdot X \tag{5}$$

We can see this as follows:

- Proof obligation (3) follows by $k_X$ uses of Lemma 2.16 (and $k_X - 1$ instances of $(\mathbf{cng}\lambda)$), since for $1 \le i \le k_X$,

$$\Delta^+ \cup \Delta_\mathcal{B} \vdash b_{Xi}\#\lambda a_{Xi+1}.\cdots\lambda a_{Xk_X}.(b_{Xi-1}\ a_{Xi-1})\cdots(b_{X1}\ a_{X1}) \cdot X.$$

- Proof obligation (4) follows by $k_X$ uses of Lemma 2.16 (and $k_X - 1$ instances of $(\mathbf{cng}\lambda)$), provided that for $1 \le i \le k_X$,

$$\Delta^+ \cup \Delta_\mathcal{B} \vdash \pi(a_{Xi})\#\lambda b_{Xi+1}.\cdots\lambda b_{Xk_X}.((\pi(a_{Xi-1})\ b_{Xi-1})\cdots(\pi(a_{X1})\ b_{X1}) \circ \pi_1) \cdot X.$$

By the rules for freshness, this follows from $\pi(a_{Xi})\#\pi_1 \cdot X \in \Delta^+ \cup \Delta_\mathcal{B}$ since the $\pi(a_{Xj})$ are all disjoint and the $b_{Xj}$ are different from $\pi(a_{Xi})$. We reason by cases on $\pi(a_{Xi})$:

- $\pi(a_{X_i}) \neq a_{X_j}$ for all $j$: then $\pi(a_{X_i})\#X \in \Delta$ by Definition 4.4.
- $\pi(a_{X_i}) = a_{X_j}$ for some $j$: then $b_{X_j}\#X \in \Delta_{\mathcal{B}}$.

- Proof obligation (5): By Theorem 2.19 it suffices to show $\Delta^+ \cup \Delta_{\mathcal{B}} \vdash \mathrm{ds}(\pi_2 \circ \pi_1, \pi)\#X$. That is, we must show that $a\#X \in \Delta^+ \cup \Delta_{\mathcal{B}}$ for every $a$ such that $(\pi_2 \circ \pi_1)(a) \neq \pi(a)$, which follows by a case distinction on $a$ (considering every $a \in \mathsf{supp}(\pi_2 \circ \pi_1) \cup \mathsf{supp}(\pi)$) using Definitions 4.4 and 4.15.

We use the properties above to construct the following ULAM derivation, presented in a calculational style:

$$
\begin{aligned}
& (\lambda e_X.e_X)\big((\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X)\pi(a_{X1})\cdots\pi(a_{Xk_X})\big) \\
= \quad & \{\ \text{Axiom } (\beta\mathbf{var})\ \} \\
& (\lambda a_{X1}.\cdots\lambda a_{Xk_X}.X)\pi(a_{X1})\cdots\pi(a_{Xk_X}) \\
= \quad & \{\ (\mathbf{cngapp}), (\mathbf{refl}), \text{and } (3)\ \} \\
& (\lambda b_{X1}.\cdots\lambda b_{Xk_X}.\pi_1 \cdot X)\pi(a_{X1})\cdots\pi(a_{Xk_X}) \\
= \quad & \{\ (\mathbf{cngapp}), (\mathbf{refl}), \text{and } (4)\ \} \\
& (\lambda \pi(a_{X1}).\cdots\lambda \pi(a_{Xk_X}).(\pi_2 \circ \pi_1) \cdot X)\pi(a_{X1})\cdots\pi(a_{Xk_X}) \\
= \quad & \{\ \text{Axiom } (\beta\mathbf{id})\ (k_X \text{ times})\ \} \\
& (\pi_2 \circ \pi_1) \cdot X \\
= \quad & \{\ (5)\ \} \\
& \pi \cdot X
\end{aligned}
$$

The result follows. □

We are now ready to prove Theorem 4.7:

**Proof (of Theorem 4.7)** Recall from Definition 4.14 the chain we fixed:

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma$$

By Lemma 4.21 and Corollary 4.23

$$\Delta^+ \vdash_{\mathsf{ULAM}} (t\varsigma)^{-1} \equiv g_1^{-1} = g_2^{-1} = \ldots = g_m^{-1} \equiv (u\varsigma)^{-1},$$

so $\Delta^+ \vdash_{\mathsf{ULAM}} (t\varsigma)^{-1} = (u\varsigma)^{-1}$ by transitivity. By Lemma 4.24 then also $\Delta^+ \vdash_{\mathsf{ULAM}} t = u$. Since $\Delta^+$ extends $\Delta$ with atoms that are not mentioned in $t$ and $u$ we extend the derivation with $(\mathbf{fr})$ to obtain $\Delta \vdash_{\mathsf{ULAM}} t = u$ as required. □

## 5 Soundness, completeness, and conservativity for $\alpha\beta\eta$

Our proof-method works for the $\lambda$-calculus augmented with $\eta$-conversion (extensionality) as well, and the proofs change only slightly. We outline how this works, emphasising only the parts of previous sections that need changing.

**Definition 5.1** Let ULAME be the nominal algebra theory with the axioms of ULAM from Figure 1, and with the additional axiom $(\eta)$, as illustrated in Figure 5.

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

**Figure 5:** Axioms of ULAME

### *5.1 Soundness*

We modify Definition 4.2 in the natural way:

**Definition 5.2** Write $\Delta \models_{\alpha\beta\eta} t = u$ when $t\sigma =_{\alpha\beta\eta} u\sigma$ (Definition 3.13) for all ground substitutions $\sigma$ for $\Delta, t, u$ such that $a \notin fa(\sigma(X))$ for every $a\#X \in \Delta$.

Soundness is just like Theorem 4.3:

**Theorem 5.3 (Soundness)** *For any $\Delta$, $t$, $u$, if $\Delta \vdash_{\mathsf{ULAME}} t = u$ then $\Delta \models_{\alpha\beta\eta} t = u$.*

**Proof** Just as for Theorem 4.3. The induction on ULAME derivations has one extra axiom case:

- If $a \notin fa(g)$ then $\lambda a.(ga) =_{\alpha\beta\eta} g$.

□

### *5.2 Completeness and conservativity*

Definition 4.4, Definition 4.5 and Lemma 4.6 are unchanged. Completeness theorem 4.7 becomes:

**Theorem 5.4 (Completeness)** *If $\Delta \models_{\alpha\beta\eta} t = u$ then $\Delta \vdash_{\mathsf{ULAME}} t = u$.*

We defer the proof to Subsection 5.3.

Corollary 4.8 becomes:

**Corollary 5.5** *For any ground terms $g, h$, $\vdash_{\mathsf{ULAME}} g = h$ if and only if $g =_{\alpha\beta\eta} h$.*

**Proof** By Theorems 5.3 and 5.4, using the fact that $g$ and $h$ are ground. □

Definition 4.9 and Lemma 4.10 are changed as follows:

**Definition 5.6** Call $g$ a **$\beta\eta$-normal form** when no $g'$ exists with $g \to_\beta g'$ or $g \to_\eta g'$.

**Lemma 5.7** *Fix $\Delta$. Suppose that $t$ and $u$ contain no subterm of the form $(\lambda a.v)w$ or $\lambda a.(va)$ where $\Delta \vdash a\#v$. Then for $\varsigma$ the ground substitution from Definition 4.5, $t\varsigma$ and $u\varsigma$ are $\beta\eta$-normal forms.*

**Proof** $\varsigma(X) \equiv e_X(d_X a_{X1} \dots a_{Xk_X})$ for every $X$ appearing in $\Delta$, $t$, $u$. Applying this substitution to $t$ and $u$ cannot introduce subterms of the form $(\lambda a.v)w$ or $\lambda a.(va)$.[11] □

[11] We chose $e_X$ in Definition 4.4, and 'wrapped' $d_X a_{X1} \dots a_{Xk_X}$ in $e_X$ in Definition 4.5, specifically to prevent accidental $\eta$-contracts here.

Lemma 4.11 is unchanged, as it does not concern ULAM. Conservativity theorem 4.12 becomes:

**Theorem 5.8 (Conservativity)** *Fix some $\Delta$. Suppose that $t$ and $u$ contain no subterm of the form $(\lambda a.v)w$ or $\lambda a.(va)$ where $\Delta \vdash a\#v$. Then*

$$\Delta \vdash_{\mathsf{ULAME}} t = u \qquad \textit{if and only if} \qquad \Delta \vdash_{\mathsf{CORE}} t = u.$$

**Proof** A derivation in CORE is also a derivation in ULAME so the right-to-left implication is immediate.

Now suppose that $\Delta \vdash_{\mathsf{ULAME}} t = u$. We construct $\varsigma$ as in Definition 4.5. By Theorem 5.3, $t\varsigma =_{\alpha\beta} u\varsigma$. By Lemma 5.7 we know that $t\varsigma$ and $u\varsigma$ are $\beta\eta$-normal forms. By confluence [Bar84, Theorem 3.3.9 (i)] $t\varsigma =_{\alpha} u\varsigma$. By Lemma 4.11 $\Delta \vdash_{\mathsf{CORE}} t = u$, as required. □

### *5.3 Proof of Theorem 5.4*

Lemma 4.13 and Definition 4.14 are changed as follows:

**Lemma 5.9** *There exists a chain*

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma$$

*where each $\leftrightarrow_?$ is one of $=_\alpha$, $\leftrightarrow_\beta$, or $\leftrightarrow_\eta$, which is such that $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for $1 \leq i \leq m$.*

**Proof** By assumption $\Delta \models_{\alpha\beta\eta} t = u$, so by Lemma 4.6 we know $t\varsigma =_{\alpha\beta\eta} u\varsigma$. It follows that there is a chain

$$t\varsigma \equiv g_1 \leftrightarrow_? g'_2 \leftrightarrow_? g'_3 \leftrightarrow_? \ldots \leftrightarrow_? g'_{m'-1} \leftrightarrow_? g_{m'} \equiv u\varsigma$$

where each $\leftrightarrow_?$ is one of $=_\alpha$, $\leftrightarrow_\beta$, or $\leftrightarrow_\eta$. By construction, $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for $i = 1$ and $i = m'$. We use part 1 *and* part 2 of Lemma 3.15, and Lemma 3.16, to construct a chain

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma$$

such that $ba(g_i) \cap (\mathcal{D} \cup \mathcal{E}) = \emptyset$ for all $1 \leq i \leq m$, just as in the proof of Lemma 4.13. □

**Definition 5.10** Make a fixed but arbitrary choice of chain as specified in Lemma 5.9.

Definitions 4.15 ($\mathcal{A}^+$) and 4.16 ($g$ accurate) are unchanged. As before (Lemma 4.17), $g_1, \ldots, g_m$ are accurate by construction. Definition 4.18 (the inverse translation $g^{-1}$) is unchanged.

Lemmas 4.19 to 4.21 do not concern ULAM, and are unchanged. Lemma 4.22 becomes:

**Lemma 5.11** *Suppose that $a \in \mathcal{A}^+ \setminus (\mathcal{D} \cup \mathcal{E})$. Suppose that $g$, $h$, and $g[h/a]$ are accurate. Then $\Delta^+ \vdash_{\mathsf{ULAME}} (\lambda a.(g^{-1}))(h^{-1}) = (g[h/a])^{-1}$.*

**Proof** By Lemma 4.22, since a derivation in ULAM is also one in ULAME. □

Corollary 4.23 changes as follows:

**Corollary 5.12** *Suppose that $g$ and $h$ are accurate.*

- *If $g \leftrightarrow_\beta h$ then $\Delta^+ \vdash_{\mathsf{ULAME}} g^{-1} = h^{-1}$.*
- *If $g \leftrightarrow_\eta h$ then $\Delta^+ \vdash_{\mathsf{ULAME}} g^{-1} = h^{-1}$.*

**Proof** The first part is by Corollary 4.23, since a derivation in ULAM is also one in ULAME.

For the second part, it suffices to show that if $a \notin fa(g)$ then $\Delta^+ \vdash_{\mathsf{ULAME}} \lambda a.(g^{-1}a) = g^{-1}$. This follows using axiom $(\eta)$ and Lemma 4.19. □

Lemma 4.24 changes as follows:

**Lemma 5.13** $\Delta^+ \vdash_{\mathsf{ULAME}} (t\varsigma)^{-1} = t$, *and* $\Delta^+ \vdash_{\mathsf{ULAME}} (u\varsigma)^{-1} = u$.

**Proof** From Lemma 4.24, since a derivation in ULAM is also one in ULAME. □

We can now prove Theorem 5.4:

**Proof (of Theorem 5.4)** Recall from Definition 5.10 the chain

$$t\varsigma \equiv g_1 \leftrightarrow_? g_2 \leftrightarrow_? g_3 \leftrightarrow_? \ldots \leftrightarrow_? g_{m-1} \leftrightarrow_? g_m \equiv u\varsigma.$$

By Lemma 4.21 and Corollary 5.12

$$\Delta^+ \vdash_{\mathsf{ULAME}} (t\varsigma)^{-1} \equiv g_1^{-1} = g_2^{-1} = \ldots = g_m^{-1} \equiv (u\varsigma)^{-1},$$

so $\Delta^+ \vdash_{\mathsf{ULAME}} (t\varsigma)^{-1} = (u\varsigma)^{-1}$ by transitivity. By Lemma 5.13 then also $\Delta^+ \vdash_{\mathsf{ULAME}} t = u$. Since $\Delta^+$ extends $\Delta$ with atoms that are not mentioned in $t$ and $u$ we extend the derivation with $(\mathbf{fr})$ to obtain $\Delta \vdash_{\mathsf{ULAME}} t = u$ as required. □

# 6 Conclusions

## *6.1 Related work*

What are functions, from a(n algebraic) logical point of view? This question has been studied from many angles:

*Lambda-lifting*
'Lambda-lifting' introduces constant symbols to represent functions and adds axioms for them [Joh85]. This expresses functions, but the axiomatisation is of atomic constant symbols representing individual functions (as many as we would like to add) and not of the $\lambda$-calculus.

The issues of variables and binding surrounding the '$\lambda$' in the '$\lambda$-calculus' are avoided — or more precisely, they are relegated to the meta-level into the universal quantifiers used in the formula expressing the properties of each atomic constant symbol. Implementationally this can be extremely convenient but from a logical point of view we should consider this deeply unsatisfactory.

*Combinatory algebra*

Schönfinkel and Curry discovered *combinatory algebra* [Sch67,CF58]. The signature contains a binary term-former *application* and two constants **S** and **K**. Axioms are $\mathbf{K}xy = x$ and $\mathbf{S}xyz = (xz)(yz)$.

This syntax is parsimonious and the axioms are compact, but it is not natural or ergonomic to program in because it is impossible to go directly from a rule specifying a function, to the function itself — by $\lambda$-abstracting. In common with lambda-lifting, this must be done at the meta-level.

There is another mathematical issue inherent to combinators themselves: the natural encoding of closed $\lambda$-terms into combinatory algebra syntax ([Bar84, Section 7] or [Sel02, Subsection 1.4]) is unsound; it does not map $\alpha\beta$-equivalent $\lambda$-terms to provably equal terms in combinatory algebra. We can strengthen combinatory algebra to *lambda algebra* by adding five more axioms [Sel02, Proposition 5] but the translation is still not sound; there exist $\lambda$-terms $M$ and $N$ such that the translation of $M$ is derivably equal to the translation of $N$, but the translation of $\lambda x.M$ is not derivably equal to the translation of $\lambda x.N$.

For soundness we need the Meyer-Scott axiom [Sel02, Proposition 20] (Selinger calls it 'the notorious rule'). Thus, combinators do not capture the same functions as expressed by the $\lambda$-calculus.

*Calculi of explicit substitutions*

Calculi of explicit substitutions capture the $\lambda$-calculus in a first-order rewrite system; the idea originates in [ACCL92] and we note also Lescanne's compact but readable survey [Les94]. If we orient the axioms of ULAM left-to-right then they become rewrite rules, and — although ULAM has no explicit substitution term-former — if we consider a $\beta$-reduct $(\lambda a.t)u$ as a(n implicit) substitution $t[a \mapsto u]$ then we can view the axioms of ULAM as 'pushing substitutions' down into a term, step-by-step. Thus, a derivation of an equality in ULAM looks very much like a sequence of explicit substitution rewrites, just not necessarily all oriented left-to-right because equality is symmetric.

ULAM displays two other notable differences from explicit substitution:

- The management of $\alpha$-equivalence (name-binding) comes from nominal techniques and is distinct from that explored in [ACCL92] or the subsequent literature. For an exploration of the computational cost of $\alpha$-equivalence in nominal terms (in the terminology of this paper; of equality up to CORE) see [Che04].
- The syntax of ULAM includes unknowns $X$, $Y$, $Z$ which explicitly represent unknown terms. 'Push a substitution down into a term' is an analogy but it cannot always be fully realised; there is no way to push $[a \mapsto X]$ into $Z$ (i.e. reduce $(\lambda a.Z)X$ using $(\beta\mathbf{id})$, $(\beta\mathbf{app})$, or $(\beta\mathbf{abs})$) because $Z$ is an unknown.

  $(\beta\#)$ resembles Bloo and Rose's 'garbage collection rule' [BR95]. There the rule is dispensable — removing it does not affect the transitive symmetric closure of the reduction relation [BR95, Remark 2.15]. In contrast, there may be models of ULAM in which $Z$ can denote an element which cannot be represented by a term. For this reason $(\beta\#)$ is not admissible in the presence of $(\beta\mathbf{id})$, $(\beta\mathbf{app})$, and $(\beta\mathbf{abs})$.

*Lambda Abstraction Algebras*

Salibra's Lambda Abstraction Algebras (LAA) [Sal00] are a $\lambda$-calculus version of cylindric algebra [HMT85]. There are many similarities with ULAM. For the convenience of the reader familiar with LAA syntax and semantics, a 'cheat-sheet' relating the material in this paper with definition 1, page 6 of [Sal00] is to the right.

| ULAM | LAA |
|---|---|
| $a, b, c$ | $x, y, z$ |
| $\lambda a, \lambda b, \lambda c$ | $\lambda x, \lambda y, \lambda z$ |
| $X, Y, Z$ | $\xi, \mu, \nu$ |
| $(\beta\mathbf{var})$ | $(\beta_\mathbf{1})$ |
| $(\beta\#)$ | $(\beta_\mathbf{2})$, $(\beta_\mathbf{4})$ |
| $(\beta\mathbf{app})$ | $(\beta_\mathbf{5})$ |
| $(\beta\mathbf{abs})$ | $(\beta_\mathbf{6})$ |
| $(\beta\mathbf{id})$ | $(\beta_\mathbf{3})$ |
| CORE | $(\alpha)$ |

ULAM exists in the nominal algebra framework; thus, at the level of syntax, there is one term-former $\lambda$ for $\lambda$-abstraction. LAA includes infinitely many term-formers $\lambda x$, $\lambda y$, $\lambda z$,...; and there are finitely many axioms and term-formers. This is because LAA uses 'ordinary' universal algebra, whereas in nominal algebra, $\alpha$-equivalence is handled primitively by the nominal algebra framework, and does not require special consideration in the signature or axioms.

Similarly, freshness side-conditions appear 'hard-wired' in LAA axioms. For example Salibra's axiom $(\beta_\mathbf{4})$ from [Sal00, page 6] $(\lambda x.(\lambda x.\xi))\mu = \lambda x.\xi$ is a version of $(\beta\#)$ where the freshness condition is built into the structure of the term by writing $\lambda x.\xi$. Nominal algebra handles freshness primitively (the 'freshness judgements' $a\#t$ of Definition 2.8). The notion of *algebraic dependence* from [Sal00, Definition 3] '$(\lambda x.a)z \neq z$ for some $z$' corresponds with the negation of our freshness judgement '$a\#X$' (in a rather unfortunate notation clash with our notation for atoms, Salibra uses $a$ to range over arbitrary elements of a model). Algebraic (in)dependence is handled rigorously but informally (that is, in the prose discourse of the paper) in Salibra's work; for us, the corresponding notion of freshness is a formal part of the syntax of nominal algebra judgements, and freshness has a denotation in nominal sets going back to the original work on nominal techniques [GP01], which places LAA 'algebraic (in)dependence' in a broader and more general mathematical setting.

For a general theory of models of ULAM, see the general theory of models of nominal algebra theories [Mat07]; recall that nominal algebra theories take models in *nominal sets*. Nominal sets have a finite-support property which, from the 'cyclindic' point of view, corresponds with a property which is known and studied under the name *locally finite* or *locally finite dimensional* [Sal00, Definition 7]. Models of ULAM are finitely-supported / locally finite by construction. [12]

See the Related Work section of [Sal00] for further references to this and similar work.

*Syntax quotiented by $\alpha\beta$-equivalence*

The definition of $\alpha\beta$-equivalence on syntax is occasionally called 'axiomatising the $\lambda$-calculus', although it is just an equivalence relation on abstract syntax trees. However, if the $\lambda$-calculus syntax serves as the language of a logic with an equality judgement then $\alpha\beta$-equivalence may have the status of axioms. For example Andrews's logic $\mathcal{Q}_0$ [And86, §51] contains five axioms $(\mathbf{4_1})$, $(\mathbf{4_2})$, $(\mathbf{4_3})$, $(\mathbf{4_4})$, and $(\mathbf{4_5})$ ([And86, page 164]). In fact these are

[12] It may be prudent to note that being finitely-supported / locally finite does not mean that in nominal sets we cannot consider models including infinitary syntax. Nominal sets are consistent with infinitary syntax, so long as it mentions only finitely many *free* variables; it can mention as many bound variables as we like, including infinitely many. For a nominal-style theory of objects with infinite support, and in particular infinitary syntax with potentially infinitely many free variables, see [Gab07].

axiom *schemes*, containing meta-variables $\mathbf{A}$ and $\mathbf{B}$ in the informal meta-level ranging over terms, and meta-variables $x$, $y$ ranging permutatively over variable-symbols. A kinship with Figure 1 is apparent, though the axioms of ULAM exist in the formal framework of nominal algebra — a formal logic, not an informal meta-level. The interested reader might like to consider a formalisation of the arguments about $\mathcal{Q}_0$ in [And86] into a formal logic (not a nominal one) [Tan05].

*Nominal techniques*

A rewrite system for the $\lambda$-calculus appeared already in [FG07b] but without any statement or proof of completeness (indeed, it was not complete). More recently the authors have used nominal algebra to axiomatise first-order logic as a theory FOL [GM08b] and substitution as a theory SUB [GM08a].

One might suspect that the theory of substitution should be only a hair's breadth away from that of the untyped $\lambda$-calculus, and could be obtained by imposing a type system (just as the simply-typed $\lambda$-calculus is related to the untyped $\lambda$-calculus). [13] Perhaps this will indeed prove to be the case, but so far all attempts by the authors to derive the properties of SUB from those of ULAM have tantalisingly failed. This may (or may not!) indicate the existence of a subtle mathematical point lurking in these systems; if there is, it seems to relate to the difference between $\lambda$-calculus variables and nominal algebra unknowns.

Nominal algebra has a cousin, nominal equational logic (**NEL**) [CP07]), which is very similar but makes slightly different design decisions. To the description of the relationship with nominal algebra given in [CP07] should be added that (unsorted) NEL is a subsystem of nominal algebra. The translation described and discussed in [GM09, Subsection 6.1.2]; it relies on the encoding of freshness using equality given in [GP01, equation 32]. NEL and nominal algebra both satisfy completeness results for a class of models in nominal sets [CP07,GM07,Mat07] (in other words — derivable equality coincides with validity *in all models*). The completeness results in this paper are stronger; they prove that derivable equality coincides with validity *in particular models*, which we built in this paper (Definitions 4.2 and 5.2). Similarly for the authors' treatments of substitution [GM08a] and first-order logic [GM08b]. We know of no like treatments of substitution, logic, and the $\lambda$-calculus in NEL. If and when this is done it will be interesting to compare the results; one way to do this is to transfer results using the translation of [GM09, Subsection 6.1.2].

*Some concluding remarks on names, variables, and functional abstraction*

Nominal techniques are based on modelling names as atoms (urelemente) in set theory [Bru96,Gab00], an idea raised by the Gabbay-Pitts models of $\alpha$-abstraction and $\mathit{И}$ quantification [GP01]. These ideas find uses beyond the initial applications to syntax: in game theory and reasoning about pointers [AGM+04,Tze07,BL05], spatial logics [LC04], and more.

Thus, there is now a body of work based on atoms, some concerned with reasoning on syntax-with-binding, some concerned with representing other things. Nevertheless, it always treats atoms as denotational entities in their own right rather than as a purely syntactic adjunct to a notion of function.

[13] We briefly give some intuition for what SUB is: $(\lambda b.(ba))(\lambda a.a) = a$ is derivable in ULAM, but only $\mathsf{app}(b,a)[b \mapsto \mathsf{lam}([a]a)] = \mathsf{app}(\mathsf{lam}([a]a), a)$ is derivable in SUB (notation from [GM08a]).

For us, nominal techniques are a *general theory of names*, and nominal algebra is a formal logic with which to instantiate this general theory to more specific theories.

$\lambda$-calculus style variables are *names* that can be *functionally abstracted*.[14] Nominal-style atoms are *names* that can be *permuted* and $\alpha$*-abstracted*. We have written down ULAM and ULAME and shown that they soundly and completely express those properties which, when added to nominal-style names, convert them into $\lambda$-calculus style variables. Conveniently, permutation then behaves like abstracted-then-applied variables (the device used, for example, in Miller's higher-order patterns [Mil91] and in Salibra's lambda-abstraction algebras [Sal00]) — and nominal-style $\alpha$-abstraction becomes $\lambda$-calculus style functional abstraction.

### *6.2 Future work and conclusions*

ULAM completes a trio of papers studying (untyped) nominal algebra considered as a logical framework: first-order logic [GM08b], substitution [GM08a], and with this paper, the $\lambda$-calculus. Between them, these works cover a significant fraction of the typical syntaxes of interest in theory and practice of computer science. It would of course be interesting to seek common generalisations of the proof-techniques therein.

Our most immediate interest is in constructing a theorem-prover based on nominal algebra instead of the $\lambda$-calculus. We expect this to formally represent at least some kinds of reasoning better than the $\lambda$-calculus can, because the treatment of names and variables in nominal algebra, and nominal techniques in general, is very close to some kinds of informal practice (for example the pervasive use of meta-variables and freshness conditions, as appear in specifications of ... the $\lambda$-calculus). From that perspective this paper proves a vital correctness result.

It remains to understand the connections between standard models for the $\lambda$-calculus and the class of models determined by models of ULAM in nominal sets. It might also be interesting to construct versions of graph models or domain models of the $\lambda$-calculus [Bar84,Sto77] that themselves are built in nominal sets; does the presence of nominal-style atoms add any useful structure? Similarly, we can consider existing work using the language of categories (for example [Sel02,AB07]) using categories based on nominal sets.

A representation theorem for ULAM, in nominal sets models, would also be interesting. As a first step, in recent work we have proved an HSP theorem (also known as Birkhoff's theorem) for nominal algebra [Gab09].

[14] This marks a difference from combinators, which share a notion of functional application with the $\lambda$-calculus but not the notion of functionally abstracting variable symbols.

# References


[AB07] Giulio Manzonetto Antonio Bucciarelli, Thomas Ehrhard. Not enough points is enough. In *Proc. 21st International Workshop on Computer Science Logic (CSL 2007)*, volume 4646 of *Lecture Notes in Computer Science*, pages 298–312. Springer, 2007.

[ABI+96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[ACCL92] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lèvy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1992.

[AGM+04] Samson Abramsky, Dan R. Ghica, Andrzej S. Murawski, C.-H. Luke Ong, and Ian D. B. Stark. Nominal games and full abstraction for the nu-calculus. In *Proc. 19th IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 150–159. IEEE Computer Society, 2004.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[Bar77] Jon Barwise. An introduction to first-order logic. In Jon Barwise, editor, *Handbook of Mathematical Logic*, pages 5–46. North Holland, 1977.

[Bar84] Henk P. Barendregt. *The Lambda Calculus: its Syntax and Semantics (revised ed.)*. North-Holland, 1984.

[Ber00] Chantal Berline. From computations to foundations via functions and application: the lambda-calculus and its webbed models. *Theoretical Computer Science*, 249(1):81–161, 2000.

[Ber06] Chantal Berline. Graph models of lambda-calculus at work, and variations. *Mathematical Structures in Computer Science*, 16(2):185–221, 2006.

[BL05] Nick Benton and Benjamin Leperchey. Relational reasoning in a nominal semantics for storage. In *Proc. 7th Int'l Conf. on Typed Lambda Calculi and Applications (TLCA)*, volume 3461 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2005.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, Great Britain, 1998.

[BR95] Roel Bloo and Kristoffer Høgsbro Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, 1995.

[Bru96] Norbert Brunner. 75 years of independence proofs by Fraenkel-Mostowski permutation models. *Mathematica Japonica*, 43:177–199, 1996.

[CDHL84] Mario Coppo, Mariangiola Dezani-Ciancaglini, Furio Honsell, and Giuseppe Longo. Extended type structures and filter lambda-models. In G. Lolli, G. Longo, and A. Marcja, editors, *Proceedings of the Logic Colloquium '82*, pages 241–262, Amsterdam, the Netherlands, 1984. North-Holland.

[CF58] Haskell B. Curry and R. Feys. *Combinatory Logic*, volume 1. North Holland, 1958.

[Che04] James Cheney. The complexity of equivariant unification. In *Proc. 31st Int'l Colloquium on Automata, Languages and Programming (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 332–344. Springer, 2004.

[Chu41] Alonzo Church. *The calculi of lambda-conversion*. Number 6 in Annals of Mathematics Studies. Princeton University Press, 1941.

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

[CP07] Ranald A. Clouston and Andrew M. Pitts. Nominal equational logic. *Electronic Notes in Theoretical Computer Science*, 172:223–257, 2007.

[dB72] N. 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.

[FG07a] Maribel Fernández and Murdoch J. Gabbay. Curry-style types for nominal terms. In *Types for Proofs and Programs (proceedings of TYPES'06)*, volume 4502 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2007.

[FG07b] Maribel Fernández and Murdoch J. Gabbay. Nominal rewriting. *Information and Computation*, 205(6):917–965, 2007.

[Fin85] Kit Fine. *Reasoning with Arbitrary Objects*. Blackwell, 1985.

[Gab00] Murdoch J. Gabbay. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, Cambridge, UK, 2000.

[Gab07] Murdoch J. Gabbay. A General Mathematics of Names. *Information and Computation*, 205(7):982–1011, July 2007.

[Gab09] Murdoch J. Gabbay. Nominal Algebra and the HSP Theorem. *Journal of Logic and Computation*, 19(2):341–367, 2009.

[GM06] Murdoch J. Gabbay and Aad Mathijssen. Nominal Algebra. In *18th Nordic Workshop on Programming Theory*, 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, 2007.

[GM08a] Murdoch J. Gabbay and Aad Mathijssen. Capture-Avoiding Substitution as a Nominal Algebra. *Formal Aspects of Computing*, 20(4-5):451–479, June 2008.

[GM08b] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order Logic. *Journal of Logic and Computation*, 18(4):521–562, August 2008.

[GM09] Murdoch J. Gabbay and Aad Mathijssen. Nominal (universal) algebra: equational logic with names and binding. *Journal of Logic and Computation*, 2009. In press.

[GP01] Murdoch J. Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects of Computing*, 13(3–5):341–363, 2001.

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

[Joh85] Thomas Johnsson. Lambda lifting: transforming programs to recursive equations. In *Conference on Functional programming languages and computer architecture*, pages 190–203. Springer, 1985.

[KNO02] Andrew D. Ker, Hanno Nickau, and C-H. Luke Ong. Innocent game models of untyped lambda-calculus. *Theoretical Computer Science*, 272(1-2):247–292, 2002.

[LC04] Luca Cardelli Luís Caires. A spatial logic for concurrency (part II). *Theoretical Computer Science*, 322(3):517–565, 2004.

[Lei94] Daniel Leivant. Higher order logic. In D. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 2, pages 229–322. Oxford University Press, 1994.

[Les94] Pierre Lescanne. From lambda-sigma to lambda-upsilon: a journey through calculi of explicit substitutions. In *Proc. 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'94)*, pages 60–69. ACM Press, 1994.

[Mat07] Aad Mathijssen. *Logical Calculi for Reasoning with Binding*. PhD thesis, Technische Universiteit Eindhoven, 2007.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 1(4):497 – 536, 1991.

[NPP08] Aleksandar Nanevski, Frank Pfenning, and Brigitte Pientka. Contextual modal type theory. *Proc. ACM Transactions on Computational Logic (TOCL)*, 9(3), 2008.

[Pau89] Lawrence C. Paulson. The foundation of a generic theorem prover. *Journal of Automated Reasoning*, 5(3):363–397, 1989.

[Pau96] Lawrence C. Paulson. *ML for the working programmer (2nd ed.)*. Cambridge University Press, 1996.

[PE88] F. Pfenning and C. Elliot. Higher-order abstract syntax. In *Proc. ACM SIGPLAN 1988 conference on Programming Language Design and Implementation (PLDI)*, pages 199–208. ACM Press, 1988.

[Sal00] Antonino Salibra. On the algebraic models of lambda calculus. *Theoretical Computer Science*, 249(1):197–240, 2000.

[Sch24] Moses Schönfinkel. Über die bausteine der mathematischen logik. *Mathematishe Annalen*, 92(3-4):305–316, 1924.

[Sch67] Moses Schönfinkel. On the building blocks of mathematical logic. In Jean van Heijenoort, editor, *From Frege to Gödel: a source book in mathematical logic, 1879-1931*. Harvard University Press, 1967. Translated from [Sch24] by Stefan Bauer-Mengelberg.

[Sel02] Peter Selinger. The lambda calculus is algebraic. *Journal of Functional Programming*, 12(6):549–566, 2002.

[Sto77] Joseph E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge, MA, USA, 1977.

[Tak95] Makato Takahashi. Parallel reductions in lambda-calculus. *Information and Computation*, 1(118):120–127, 1995.

[Tan05] Li-Yang Tan. Formalizing the meta-theory of $\mathcal{Q}_0$ in Rogue-Sigma-Pi. In *17th European Summer School in Logic, Language and Information (ESSLLI), student session*, 2005.

[Tho96] Simon Thompson. *Haskell: The Craft of Functional Programming*. Addison Wesley, 1996.

[Tze07] Nikos Tzevelekos. Full abstraction for nominal general references. In *Proc. 22th IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 399–410. IEEE Computer Society, 2007.

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

$$\frac{}{a \approx_\Delta a}\ (\mathbf{Ax}) \qquad \frac{\Delta \vdash \mathrm{ds}(\pi,\pi')\#X}{\pi \cdot X \approx_\Delta \pi' \cdot X}\ (\mathbf{Ds}) \qquad \frac{t' \approx_\Delta u' \quad t \approx_\Delta u}{t't \approx_\Delta u'u}\ (\mathbf{Fapp}) \qquad \frac{}{\mathsf{c} \approx_\Delta \mathsf{c}}\ (\mathbf{Fc})$$

$$\frac{t \approx_\Delta u}{\lambda a.t \approx_\Delta \lambda a.u}\ (\mathbf{Absaa}) \qquad \frac{(b\ a) \cdot t \approx_\Delta u \quad \Delta \vdash b\#t}{\lambda a.t \approx_\Delta \lambda b.u}\ (\mathbf{Absab})$$

**Figure A.1:** Syntax-directed rules for CORE

# A Syntactic criteria for CORE-equality

For completeness we include a proof of Theorems 2.19. Note that this is a basic result of theory CORE and hence of nominal algebra in general; it does not have to do with the $\lambda$-calculus, or with ULAM.

We will show that theory CORE is equivalent to an existing notion of $\alpha$-equivalence on nominal terms, which is syntax-directed by definition [UPG04,FG07b]. Definition A.1 was introduced in [UPG04, Figure 2] (except that here, our syntax is specific to the $\lambda$-calculus):

**Definition A.1** Let $t \approx_\Delta u$ be an ordered tuple of a term $t$, a freshness context $\Delta$, and a term $u$. Let the **derivable equalities of** $t \approx_\Delta u$ be inductively defined by the rules in Figure A.1.

**Lemma A.2** *$(\pi' \circ \pi) \cdot t \equiv \pi' \cdot (\pi \cdot t)$ and $id \cdot t \equiv t$.*

**Proof** By routine inductions on $t$. The only interesting case is $t \equiv \pi'' \cdot X$ for the first part: by Definition 2.5 we can verify that

$$(\pi' \circ \pi) \cdot (\pi'' \cdot X) \equiv (\pi' \circ \pi \circ \pi'') \cdot X \quad \text{and} \quad \pi' \cdot (\pi \cdot (\pi'' \cdot X)) \equiv (\pi' \circ \pi \circ \pi'') \cdot X.$$

□

Recall the notion of difference set $\mathrm{ds}(\pi,\pi')$ of two permutations $\pi$ and $\pi'$ from Definition 2.18. Lemma A.3 shows how theory CORE can mimic the $(\mathbf{Ds})$ rule of $\approx_\Delta$:

**Lemma A.3** *If $\Delta \vdash \mathrm{ds}(\pi,\pi')\#t$ then $\Delta \vdash_{\mathsf{CORE}} \pi \cdot t = \pi' \cdot t$.*

**Proof** We work by induction on the number of elements in $\mathrm{ds}(\pi,\pi')$. If this set is empty then $\pi = \pi'$ and the result follows easily by $(\mathbf{refl})$. Now suppose $a \in \mathrm{ds}(\pi,\pi')$. We construct a partial derivation of the proof obligation:

$$\frac{\pi \cdot t = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot t \qquad \dfrac{\pi(a)\#\pi' \cdot t \quad \pi'(a)\#\pi' \cdot t}{((\pi(a)\ \pi'(a)) \circ \pi') \cdot t = \pi' \cdot t}\ (\mathbf{perm})}{\pi \cdot t = \pi' \cdot t}\ (\mathbf{tran})$$

(The instance of $(\mathbf{perm})$ here is valid because by Lemma A.2 $((\pi(a)\ \pi'(a)) \circ \pi') \cdot t \equiv ((\pi(a)\ \pi'(a)) \cdot (\pi' \cdot t)$, and $\pi' \cdot t \equiv id \cdot (\pi' \cdot t)$.)

The following proof obligations remain:

- $\pi \cdot t = ((\pi(a)\ \pi'(a)) \circ \pi') \cdot t$ follows from $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')\#t$ by the inductive hypothesis, provided that $|\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')| < |\mathrm{ds}(\pi, \pi')|$. This condition is satisfied, since $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi') = \mathrm{ds}(\pi, \pi') \setminus \{a\}$. The remaining proof obligation $\mathrm{ds}(\pi, (\pi(a)\ \pi'(a)) \circ \pi')\#t$ follows by assumption $\mathrm{ds}(\pi, \pi')\#t$.
- $\pi(a)\#\pi' \cdot t$ follows from $(\pi')^{-1}(\pi(a))\#t$ by an easy calculation. Now this is one of the assumptions $\mathrm{ds}(\pi, \pi')\#t$: by Definition 2.18, we know that $(\pi')^{-1}(\pi(a)) \in \mathrm{ds}(\pi, \pi')$ when $\pi((\pi')^{-1}((\pi(a))) \neq \pi(a)$, and, using the fact that $\neq$ is invariant under permutation, this follows from the assumption $\pi(a) \neq \pi'(a)$.
- $\pi'(a)\#\pi' \cdot t$ follows from $a\#t$ by an easy calculation. This follows directly from assumption $\mathrm{ds}(\pi, \pi')\#t$, since $a \in \mathrm{ds}(\pi, \pi')$.

□

**Theorem A.4 (Equivalence of CORE and $\approx_\Delta$)** *$\Delta \vdash_{\mathsf{CORE}} t = u$ is derivable if and only if $t \approx_\Delta u$ is derivable using the rules of Figure A.1.*

**Proof** The left-to-right direction is by induction on the structure of nominal algebra derivations of $\Delta \vdash_{\mathsf{CORE}} t = u$. By the inductive hypothesis it suffices to show the following:

- Syntax-directed equality $\approx_\Delta$ is an equivalence relation and a congruence. This is [FG07b, Theorem 24].
- If $\Delta \vdash a\#t$ and $\Delta \vdash b\#t$ then $(a\ b) \cdot t \approx_\Delta t$. By induction on $t$.
- If $t \approx_{\Delta, a\#X_1, \ldots, a\#X_n} u$ where $a \notin t, u, \Delta$ then $t \approx_\Delta u$. By straightforward induction on the structure of derivations of $t \approx_{\Delta, a\#X_1, \ldots, a\#X_n} u$.

For the right-to-left direction we work by induction on derivations of $t \approx_\Delta u$. By the inductive hypothesis it suffices to show:

- $\Delta \vdash_{\mathsf{CORE}} a = a$. This is an instance of (**refl**).
- If $\Delta \vdash \mathrm{ds}(\pi, \pi')\#X$ then $\Delta \vdash_{\mathsf{CORE}} \pi \cdot X = \pi' \cdot X$. This is Lemma A.3.
- If $\Delta \vdash_{\mathsf{CORE}} t' = u'$ and $\Delta \vdash_{\mathsf{CORE}} t = u$ then $\Delta \vdash_{\mathsf{CORE}} t't = u'u$. This is (**congapp**).
- $\Delta \vdash_{\mathsf{CORE}} \mathsf{c} = \mathsf{c}$. This is an instance of (**refl**).
- If $\Delta \vdash_{\mathsf{CORE}} t = u$ then $\Delta \vdash_{\mathsf{CORE}} \lambda a.t = \lambda a.u$. This is (**cong**$\lambda$).
- If $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ then $\Delta \vdash_{\mathsf{CORE}} \lambda a.t = \lambda b.u$. Suppose that $\Pi$ and $\Pi'$ are derivations of $\Delta \vdash_{\mathsf{CORE}} (b\ a) \cdot t = u$ and $\Delta \vdash b\#t$ respectively. Then the following is a derivation of $\Delta \vdash_{\mathsf{CORE}} \lambda a.t = \lambda b.u$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ \Pi'}{b\#t}\,(\#\lambda\mathbf{b})}{b\#\lambda a.t} \qquad \dfrac{}{a\#\lambda a.t}\,(\#\lambda\mathbf{a})}{\lambda b.(b\ a) \cdot t = \lambda a.t}\,(\mathbf{perm})}{\lambda a.t = \lambda b.(b\ a) \cdot t}\,(\mathbf{symm}) \qquad \dfrac{\vdots\ \Pi}{(b\ a) \cdot t = u}}{\lambda b.(b\ a) \cdot t = \lambda b.u}\,(\mathbf{cong}\lambda)$$
$$\lambda a.t = \lambda b.u\ (\mathbf{tran})$$

□

**Proof (of Theorem 2.19)** From Theorem A.4, since the rules in Figure A.1 render the criteria of Theorem 2.19 as derivation rules. □