Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 298 (2013) 93–117

www.elsevier.com/locate/entcs

Nominal Lambda Calculus: An Internal Language for FM-Cartesian Closed Categories

Roy L. Crole and Frank Nebel

Dept of Computer Science, University of Leicester, University Road, Leicester, LE1 7RH, U.K.

Abstract

Reasoning about atoms (names) is difficult. The last decade has seen the development of numerous novel techniques. For equational reasoning, Clouston and Pitts introduced Nominal Equational Logic (NEL), which provides judgements of equality and freshness of atoms. Just as Equational Logic (EL) can be enriched with function types to yield the lambda-calculus (LC), we introduce NLC by enriching NEL with (atom-dependent) function types and abstraction types. We establish meta-theoretic properties of NLC; define ИFM-cartesian closed categories, hence a categorical semantics for NLC; and prove soundness & completeness by way of NLC-classifying categories. A corollary of these results is that NLC is an internal language for ИFM-cccs. A key feature of NLC is that it provides a novel way of encoding freshness via dependent types, and a new vehicle for studying the interaction of freshness and higher order types.

Keywords: category theory, dependent types, FM-sets, internal language, nominal logic, semantics, type theory

1 Introduction

(NEL) was introduced by Clouston and Pitts in [8] (closely related to Nominal Algebra introduced by Gabbay and Mathijssen [16]). Space forces us to assume familiarity with NEL, but here is a quick overview: NEL extends equational logic EL [10,24] (where types denote ZF-sets). NEL variables are thought of as elements of FM-sets (roughly speaking, sets whose elements have a finite support of atoms/(names) in the sense of Gabbay and Pitts [14] and for which one can make assertions about the freshness of atoms). The motivation for NEL is to provide a system for formal equational reasoning *combined* with reasoning about the freshness of atoms—the latter an important topic of study in Programming Semantics. To this end one seeks a theory with a sound and complete semantics. The theory must necessarily capture permutation actions, finite support, and freshness. As such, one might expect to be able to make judgements $a \# M$, asserting atom a is

[1] Email: r.crole@mcs.le.ac.uk

http://dx.doi.org/10.1016/j.entcs.2013.09.009

fresh for M, as well as $M = M'$. Further, we need to be able to assert hypotheses $a \# x$ about variables x that may occur (freely) in M. Indeed in NEL one sees $\overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n \vdash^{\#} \overline{a} \# M : s$ capturing the intuition that if sets of atoms $\overline{a}_i$ are fresh for (the interpretation of) the x_i, then $\overline{a}$ is fresh for M. One might also work instead with judgements $\overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n \vdash^{E} M : s$ and then codify $\overline{a} \# M$ by way of an equation, since freshness can be defined equationally [6] (under suitable conditions). This is the approach we take.

Clouston has shown in [5,7] that the category *FMSet* provides a sound and complete semantics for NEL. Further he defines the notion of an FM-category, axiomatising the underlying structure of *FMSet*, and shows that such categories yield a sound semantics. He shows that a NEL theory has a classifying FM-category in which there is a generic model [10,31]—hence his semantics is also complete. Indeed, Clouston shows that there is a correspondence between NEL theories and FM-categories establishing that NEL is an internal language for FM-categories. Lambek [21] showed that theories in the λ-calculus correspond to cartesian closed categories (a proof using functional completeness, with Scott, appears in [22]; see also [10]).

A natural question to ask is whether there is a notion of nominal λ-calculus (NLC) that corresponds to some form of "cartesian closed FM-category". Moreover, if there is, we can test the robustness of both NEL, and the methodology of categorical logical relations, by attempting to show that NLC is conservative over NEL using gluing. To do this we need to develop NLC and a suitable categorical correspondence, which we do in this paper.

Before we begin the task at hand, we justify our overall approach. At the conceptual level, this paper concerns itself with the fascinating notion of correspondences between category theory and type theory. This arises from Lawvere's seminal work [23]. There are two approaches that one could take in formulating such correspondences. (i) is to demonstrate that models of a theory Th in a category $\mathcal{C}$ (and maps between models), $Mod(Th, \mathcal{C})$, correspond to structure preserving functors (and natural transformations between functors), $SPF(Cl(Th), \mathcal{C})$. (ii) is to show the existence of a monad T_{Th} for which $Mod(Th, \mathcal{C})$ corresponds to the (Eilenberg-Moore) algebras of T_{Th}. Both approaches have their merits. For some deep insights into the heart of the matter in the case of theories in equational logic consult Hyland and Power's overview [19]. An elegant approach via monads, providing a very general framework, is established in the work of Berger, Mèlliés and Weber [2] and Mèlliés [26]. However, for computer science and (foundations of) program semantics, where one may well be seeking a rigorously specified syntactic type theory capable of being formalised, approach (i) seems to be the path to follow (please see Section 7 for additional commentary). In particular, we want to establish that any such theory is indeed the internal language of a suitable category with structure, with the usual adjunction $Cl \dashv Th$.

Remark 1.1 The category central to this work is *FMSet* [5,15]. The category of nominal sets *FMNom* is relevant too: for a very clear introduction see [30]. While the properties of *FMSet* are less well known than *FMNom*, both are toposes $\mathcal{T}$.

As such each is equipped with a Higher Order Logic internal language $Th_{\mathcal{T}}$. Thus one might ask whether one could automatically capture the notion of FM-ccc by internalisation of cartesian closure (and freshness) in $\mathcal{T}$; and indeed "extract" NLC from the HOL $Th_{\mathcal{T}}$, perhaps by extending $Th_{\mathcal{T}}$ with additional axioms. It is not clear to us that this can be done, or, if it can, whether it it can circumvent the detail in this paper bearing in mind that our aim from the "computer science" perspective is to produce a fully formalised type theory. See Section 7 for more discussion.

We build directly on [7], taking approach (i). We have tried to keep this paper as self-contained as possible, but cannot include all of the definitions and lemmas for lack of space. In Section 2 we specify the types and terms of NLC without abstraction. We define permutation actions, capture avoiding substitution, and α-equivalence. We prove results about the terms which we will use when proving soundness and completeness. In Section 3 we specify the NLC type system and define NLC equational theories, without abstraction. We again prove key results for soundness and completeness. In Section 4 we introduce FM-cartesian closed categories, showing they soundly model NLC without abstraction. In Section 5 we add abstraction and concretion to NLC and show that our semantics is sound and complete for ИFM-cccs, which are FM-cccs with additional structure that models abstraction and concretion. In Section 6 we show that ИFM-cccs are syntax free presentations of NLC theories. In Section 7 we discuss applications and further work. Here are the main contributions:

- Higher order functions that naturally extend NEL are partial in the sense that their arguments must satisfy freshness conditions. We believe that this is the first paper to posit a move to a "types dependent on atoms" type theory in order to capture, in a novel type system, this partiality of higher order functions (see page 4 for details). NLC allows us to examine the combination of the freshness relation and higher types in a new light.
- Dependent types enable us to to specify name abstraction and concretion. The operation of concretion is inherently partial, and indeed cannot be captured as a NEL theory—see Clouston [4]. However NLC dependent types do provide a mechanizm to capture this partiality.
- In [7] the type system for terms is separate from the system for freshness assertions, (a two part type system). Moreover typing judgements predicated on freshness assertions are not first class citizens (but simply reflexive equations). We introduce rules for a single first class type system. This is not only necessitated by the dependent types, but significantly simplifies and unifies the judgements forms in [7].
- A clean formulation of a categorical semantics of NLC. The semantics is considerably complicated by both type dependency on atoms, and the encapsulation of freshness judgements by equational axioms. Our single first class type system simplifies our soundness proof from what it would otherwise have been.
- A simplification of Clouston's meta-theory [7]. We show that all the key properties of (syntactic) permutation actions we require can be defined cleanly on raw terms,

prior to type-checking. This material is mainly in Section 2.

- A detailed proof of an "approach (i)" category theory type theory correspondence, yielding NLC theories as the internal language of FM-cccs, and hence completeness. We pay very careful attention to details that are significant for implementations (see for example the proof on page 11 of Lemma 3.3).

We will use the following notation: Let $\mathbb{A}$ be the set of atoms (names). We write $\overline{a}$ or similar for typical finite subsets $\{a_1, \ldots, a_k\}$ of $\mathbb{A}$. We write π or similar for any permutation on $\mathbb{A}$ with finite domain. *Perm* denotes the set of such permutations (equivalently those generated by transpositions $(a\,b)$). The composition of π and π', with π' acting first, is denoted by $\pi \circ \pi'$ or $\pi\pi'$. If $X = (X, \cdot)$ is an FM-set, and $x \in X$, we write $supp(x)$ for the support of x, and $\overline{a} \# x$ to denote that each atom in $\overline{a}$ is not in $supp(x)$.

2 The Meta-Theory of NLC Terms without Abstraction

Remark 2.1 Until Section 5 we work with a subset of NLC. This will allow us to fully motivate the use of a form of dependent typing in order to formulate our extension of NEL with higher order functions. Abstraction and concretion is omitted until later in the paper.

In NEL one works with a nominal set of types[2]. In NLC we work with a nominal set of ground types, and generate the function types. NLC extends NEL terms with function abstractions and applications. An abstraction takes the form $\lambda^{\overline{a}} x : s.\, M$ and we explain the intended semantic interpretation. In NEL we may have $\overline{a} \# x : s \vdash^E M : s'$. If we want to capture the "mapping" $x \mapsto M$ as an abstraction, we could consider $\lambda\, x : s.M$. However, if we apply $\lambda\, x : s.M$ to a term $N : s$ we also need to ensure that $\overline{a} \# N$. We might codify the set $\overline{a}$ in the abstraction $\lambda^{\overline{a}} x : s.\, M$. So far so good. But what about types? In NEL, the FM-set semantics of $\overline{a} \# x : s$ is specified by requiring that $[\![x]\!] \in [\![s]\!]^{\#\overline{a}} \stackrel{\text{def}}{=} \{e \in [\![s]\!] \mid \overline{a} \# e\}$. So one might wonder if $s^{\overline{a}}$ could be be a suitable type for the source of our abstraction, with a compositional semantics $[\![s^{\overline{a}}]\!] \stackrel{\text{def}}{=} [\![s]\!]^{\#\overline{a}}$. We can then consider the type $s^{\overline{a}} \Rightarrow s'$ for our abstraction, hoping that if our semantics is defined in a compositional way, it will have all of the relevant equivariance and categorical properties to yield a sound and complete semantics. This abstraction typing is deceptively simple: the type and equation system that results is intuitive, but quite complex to manipulate since function types now depend on atoms.

NLC-*Signatures, Types, and Raw Terms.* We start with an analogue of the notion of a signature for λ-calculus. A NLC-signature *Sg* is specified by

(i) Gnd_{Sg}, a nominal **set of ground types**. The **set of types** $Type_{Sg}$ is then generated by the BNF grammar $s ::= \gamma \mid s^{\overline{a}} \Rightarrow s$ where γ is any ground type. Since each type s is a finite tree and Gnd_{Sg} is a nominal set, each s is finitely

[2] In [7] "types" are called sorts. We use the word type since it better matches general usage in computer science, and categorical type theory

supported with the permutation action

$$\pi \cdot \gamma \stackrel{\text{def}}{=} \pi \cdot_{Gnd_{Sg}} \gamma \qquad \pi \cdot (s^{\overline{a}} \Rightarrow s') \stackrel{\text{def}}{=} (\pi \cdot s)^{\pi \cdot \overline{a}} \Rightarrow (\pi \cdot s')$$

and hence $Type_{Sg}$ is a nominal set of types.

(ii) A nominal **set of (higher order function) constant symbols** Fun_{Sg}.

(iii) An equivariant typing function $Fun_{Sg} \to Type_{Sg}$, which assigns to each constant symbol c a type; we refer to a **typing** $c : s$.

Fixing a set $Var \stackrel{\text{def}}{=} \{\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3, \ldots\}$ of (ordered) **variables**, the raw NLC-terms are specified by $M := \pi x \mid c \mid \lambda^{\overline{a}} x : s.\, M \mid M\ M$ where πx is a **suspension** [8,7] of any variable $x \in Var$. We refer to the set of raw terms for signature Sg by $Term_{Sg}$. Variables may be **free** or **bound** (where all occurrences of x in any "subterm" $\lambda^{\overline{a}} x : s.\, M$ are bound).

Permutation Actions for Raw Terms. Recall [8,16] the two standard permutation actions on $Perm$, namely conjugation (which is finitely supported) and left multiplication (which is not). Clouston & Pitts and Gabbay & Mathijssen define two permutation actions, called meta-level $\pi \cdot M$ and object-level $\pi * M$ [8,7], which are syntactic analogues of the actions on $Perm$. In categorical type theory one always works with terms in context. As such, a term M with a free variable x is always regarded as a "function" $x \mapsto M$. The permutation action on functions found in nominal and FM-sets is a (form of) conjugation action and the syntactic analogue is $\pi \cdot M$. However it is useful to work also with a simple action in which π acts on M simply by acting recursively over the structure of a term: eg $\pi * (\tau x)(\tau' y) = (\pi * \tau x)(\pi * \tau' y) = (\pi\tau x)(\pi\tau' y)$.

We define such actions for NLC. To do so, consider the recursive definition of mappings $(\pi, M) \mapsto \pi * M$ and $(\pi, M) \mapsto \pi \cdot M$ in Table 1. Note that in order to define the object-level permutation we first define a basic form of substitution $M[\pi^{-1}x/x]$, on raw terms M. We call this a **suspension-substitution**. Informally, free occurrences of x in M are replaced by $\pi^{-1}x$. Formally, the recursive definition is the expected one, where on suspensions we define $(\pi' y)[\pi^{-1}x/x] \stackrel{\text{def}}{=} \pi' y$ if $x \neq y$ and $(\pi' x)[\pi^{-1}x/x] \stackrel{\text{def}}{=} (\pi'\pi^{-1})x$.

To show, in Proposition 2.3, that the mappings in Table 1 are permutation actions, we need Lemma 2.2 which is proved by induction over M.

Lemma 2.2 $(\pi * M)[\pi^{-1}x/x] = \pi * (M[\pi^{-1}x/x])$ *for any raw* M*, where* $[\pi^{-1}x/x]$ *indicates that* x *is replaced by* $\pi^{-1}x$*.*

Proposition 2.3 (Permutation Action Definitions)

- *The mapping* $(\pi, M) \mapsto \pi \cdot M$ *is a permutation action; we call it the* ***meta-level*** *permutation action. It is finitely supported, so the set* $Term_{Sg}$ *of raw* NLC*-terms is a nominal set. The finite support of a raw term is specified recursively where* $supp(\pi x) \stackrel{\text{def}}{=} supp(\pi)$*,* $supp(\lambda^{\overline{a}} x : s.\, M) \stackrel{\text{def}}{=} \overline{a} \cup supp(s) \cup supp(M)$ *and* $supp(M\ N) \stackrel{\text{def}}{=} supp(M) \cup supp(M)$*; and constants are finitely supported by defi-*

• $\pi \cdot \pi' x \stackrel{\text{def}}{=} (\pi\pi'\pi^{-1})x$	• $\pi * \pi' x \stackrel{\text{def}}{=} (\pi\pi')x$
• $\pi \cdot c \stackrel{\text{def}}{=} \pi \cdot_{Fun_{Sg}} c$	• $\pi * c \stackrel{\text{def}}{=} \pi \cdot_{Fun_{Sg}} c$
• $\pi \cdot (\lambda^{\overline{a}} x : s.M) \stackrel{\text{def}}{=} \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.(\pi \cdot M)$	• $\pi * (\lambda^{\overline{a}} x : s.M) \stackrel{\text{def}}{=} \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.(\pi * (M[\pi^{-1}x/x]))$
• $\pi \cdot (M\ N) \stackrel{\text{def}}{=} (\pi \cdot M)\ (\pi \cdot N)$	• $\pi * (MN) \stackrel{\text{def}}{=} (\pi * M)(\pi * N)$
Meta-Level	Object-Level

Table 1
Permutation Actions for NLC

nition.

- *The mapping* $(\pi, M) \mapsto \pi * M$ *is a permutation action; we call it the* ***object-level*** *permutation action.*

Capture Avoiding Substitution and α*-Equivalence.* We require simultaneous capture-avoiding substitution of raw terms. This will be crucial for defining composition of morphisms in a classifying category–see Proposition 5.1. Since the high level ideas of this paper can be read without recourse to complete detail, we just outline our notation and the key ideas (our approach simplifies Clouston's [7]). Substituting $N_1, \ldots, N_n$ for free occurrences of the distinct variables $x_1, \ldots, x_n$ in the raw term M yields another raw term, which we denote by $M\{N_1, \ldots, N_n/x_1, \ldots, x_n\}$ or by $M\{\boldsymbol{N_i}/\boldsymbol{x_i}\}$. The "usual" recursive definition for "ordinary" λ-terms (see, for example, [18]) carries over to NLC apart from the base cases on suspensions where we define

$$(\pi y)\{N_1, \ldots, N_n/x_1, \ldots, x_n\} =_{def} \pi y \qquad (\forall i)(x_i \neq y)$$
$$(\pi y)\{N_1, \ldots, N_n/x_1, \ldots, x_n\} =_{def} \pi * N_{i_0} \qquad (\exists i)(x_i = y) \text{ with } x_{i_0} = y$$

Note the critical use of the object-level permutation action. Note also the crucial connection—used in many proofs—between suspension-substitutions and simultaneous substitution, which is easily proved by induction:

Lemma 2.4 *For any term* M *we have* $M[\pi^{-1}x/x] = M\{\pi^{-1}x/x\}$.

So far we have used structural equality on terms $M = N$. Since we wish to work with capture avoiding substitution (to construct our classifying category) which makes use of variable renaming, we have to replace $=$ with α-equivalence $\sim_\alpha$. We use two definitions of α-equivalence. One is founded on capture avoiding substitution; the other on variable swapping. Each definition generates the same relation $\sim_\alpha \subset Term_{Sg} \times Term_{Sg}$ (see [11]).

The first definition [18] takes $\sim_\alpha$ to be the smallest equivalence relation closed under the congruence rules (for application and abstraction terms) and the axiom $\lambda^{\overline{a}} x : s.\, M \sim_\alpha \lambda^{\overline{a}} x' : s.\, M\{x'/x\}$ where $x' \notin var(M)$. The second definition is given

$$\frac{}{\pi x \sim_\alpha \pi x}\ (x \in Var \quad \pi \in Perm) \qquad \frac{M_1 \sim_\alpha M_1' \qquad M_2 \sim_\alpha M_2'}{M_1 M_2 \sim_\alpha M_1' M_2'}$$

$$\frac{(z\,x) \bullet M_1 \sim_\alpha (z\,y) \bullet M_2}{\lambda^{\overline{a}} x : s.\, M_1 \sim_\alpha \lambda^{\overline{a}} y : s.\, M_1}\ (z \notin var(M_1) \cup var(M_2))$$

Table 2
Alpha Equivalence by Variable Swapping

in terms of variable swapping [11,14]. If $x, y \in Var$ then we define $(x\,y) \bullet M$ to be M in which any occurrence of x is swapped with y (and vice-versa). Then we can define $\sim_\alpha$ by the rules in Table 2. It can easily be shown that $\sim_\alpha$ is equivariant for the permutation actions, that is $M \sim_\alpha N$ implies $\pi \cdot M \sim_\alpha \pi \cdot N$ and respectively for the object level permutation action. From this well-defined permutation actions on α-equivalence classes of terms are induced by way of the following definitions $\pi \cdot [M]_\alpha \stackrel{\text{def}}{=} [\pi \cdot M]_\alpha$ and $\pi * [M]_\alpha \stackrel{\text{def}}{=} [\pi * M]_\alpha$ and moreover we can prove

Lemma 2.5 *Capture avoiding substitution lifts to the set of α-equivalence classes of terms, ${}^{Term_{Sg}}/_{\sim_\alpha}$, a nominal set under the meta-level permutation action on α-equivalence classes, with $supp([M]_\alpha) = supp(M)$.*

Remark 2.6 We call $[M]_\alpha$ an **expression**. Having taken great care in defining expressions $[M]_\alpha$, we adopt the usual convention of writing just M. However, all our proofs deal correctly with the intricacies that arise from variable re-naming to avoid capture (see for example [28] (page 169) and [25]).

The next propositions are crucial for our main theorems, the first ($*$ associates with $\{/\}$) by induction on M, the second ($*$ distributes over $\{/\}$) by direct calculation being a corollary of Proposition 2.8 and Lemma 2.7. The lemma expresses the meta-level action in terms of the object-level action. In fact it is not only used to prove properties of NLC but also, later on, our categorical semantics.

Lemma 2.7 ($\cdot$ in terms of $*$) *For any term M and $\{x_1, ..., x_n\} \subseteq Var$ with $fv(M) \subseteq \{x_1, ..., x_n\}$ we have $\pi \cdot M = (\pi * M)\{\pi^{-1}x_1/x_1, ..., \pi^{-1}x_n/x_n\}$.*

Proof. Although the proof of this lemma is straightforward, since it is quite typical we give full details of the proof by induction on the structure of term M of

$$(\forall \pi)(\forall \{x_1, \ldots, x_n\})(fv(M) \subseteq \{x_1 \ldots x_n\} \\ \Longrightarrow \pi \cdot M = (\pi * M)\{\pi^{-1}x_1/x_1, ..., \pi^{-1}x_n/x_n\}$$

We assume Lemma 2.4 throughout.

SUSP: When M is τx_i the result follows immediately by the definition of substitution and the permutation actions. **CONST:** Follows immediately. **APP:** Straightforward.

LAM-ABS: Case M is $\lambda^{\overline{a}} x : s.\, M'$ where $fv(\lambda^{\overline{a}} x : s.\, M') \stackrel{\text{def}}{=} fv(M') \setminus \{x\} \subseteq \{x_1 \ldots x_n\}$. We examine the case when x is not an x_i; if x is an x_i the details are

not too dissimilar. So for the induction step $fv(M') \subseteq \{x, x_1 \ldots, x_n\}$.

$$
\begin{aligned}
&\pi \cdot (\lambda^{\overline{a}} x : s.M') \\
&\quad \stackrel{\text{def}}{=} \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.\pi \cdot M' \\
&\quad = \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.(\pi * M')\{\pi^{-1}x/x, \boldsymbol{\pi^{-1}x_i/x_i}\} && \text{(induction)} \\
&\quad = \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.((\pi * M')\{\pi^{-1}x/x\})\{\boldsymbol{\pi^{-1}x_i/x_i}\} && (x \neq x_i) \\
&\quad = \lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.(\pi * (M'\{\pi^{-1}x/x\}))\{\boldsymbol{\pi^{-1}x_i/x_i}\} && \text{(Lemma 2.2)} \\
&\quad = (\lambda^{\pi\cdot\overline{a}} x : \pi \cdot s.\pi * (M'\{\pi^{-1}x/x\}))\{\boldsymbol{\pi^{-1}x_i/x_i}\} && (x \neq x_i \text{ so no capture}) \\
&\quad \stackrel{\text{def}}{=} (\pi * (\lambda^{\overline{a}} x : s.M'))\{\boldsymbol{\pi^{-1}x_i/x_i}\}
\end{aligned}
$$

□

For expressions $[M]_\alpha$, distinct variables $x_1, \ldots, x_n$, and expressions $[N_1]_\alpha, \ldots, [N_n]_\alpha$ we have

Proposition 2.8 $(\pi * [M]_\alpha)\{[\boldsymbol{N_i'}]_{\boldsymbol{\alpha}}/\boldsymbol{x_i}\} = \pi * ([M]_\alpha\{[\boldsymbol{N_i'}]_{\boldsymbol{\alpha}}/\boldsymbol{x_i}\})$

Proposition 2.9 $\pi \cdot (M\{\boldsymbol{N_i}/\boldsymbol{x_i}\}) = (\pi \cdot M)\{\boldsymbol{(\pi \cdot N_i)}/\boldsymbol{x_i}\}$ *(Written using the convention for α-equivalence classes, generally adopted from now on.)*

3 NLC Typed Expressions and Equational Theories

We define NLC by specifying a type and equation system. The intuitions of NLC and NEL are the same, but technicalities are quite different. In NEL, terms are typed using environments $\Gamma \stackrel{\text{def}}{=} x_1 : s_1, \ldots, x_n : s_n$, just like ordinary equational logic. The judgements either take the form $\Gamma \vdash M : s \quad (\diamond)$, or $\nabla \vdash^E M \approx M' : s$ where $\nabla = \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$ records assumptions about freshness and types. NEL judgements $\nabla \vdash^E M : s$ are simply sugar for reflexive equations. The type system $(\diamond)$ is entirely separate from the freshness system (in two parts)! We found this slightly confusing. Indeed, with NLC we cannot separate the type system in this way, since the types of abstractions depend directly on freshness assertions. Thus the environments used in the type system must encode freshness assertions (and cannot be of the form Γ)! Our typing judgements $\nabla \vdash^E M : s$ are first class citizens (in a single system). They are not abbreviations for reflexive equations. This is not merely dabbling with unnecessary cosmetic idolatry: it simplifies the presentation of our categorical semantics and is a key contribution.

Recall the formal notion of a freshness environment [7] (included below). We can then define expressions, and equations, in context and finally present the NLC type and equation systems.

A **freshness environment**, or just **environment**, is a finite partial function $\nabla : Var \to \mathcal{P}_{fin}(\mathbb{A}) \otimes Type_{Sg}$ with finite domain. By definition it maps each $x \in dom(\nabla)$ to a pair $(\overline{a}, s)$ where $\overline{a}$ is a finite set of atoms $s \in Type_{Sg}$ and $\overline{a} \# s$. The set of environments Env_{Sg} is a nominal set under the permutation action $(\pi \cdot \nabla)(x) = (\pi \cdot \overline{a}, \pi \cdot s)$. We often write an environment ∇ as $\overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$. For ∇, ∇', we write $\nabla \leq \nabla'$ if $dom(\nabla) \subseteq dom(\nabla')$ and for all $x \in dom(\nabla)$ we have $pr_1(\nabla(x)) \subseteq pr_1(\nabla'(x))$ and $pr_2(\nabla(x)) = pr_2(\nabla'(x))$.

$$(\mathsf{SP})\quad \frac{}{\nabla, \overline{a} \# x : s \vdash^E \pi x : \pi \cdot s}$$

$$(\mathsf{C})\quad \frac{}{\nabla \vdash^E c : s}\ (c \in Fun_{Sg} \text{ and } c \text{ has } Sg \text{ typing } c : s)$$

$$(\mathsf{ABS})\quad \frac{\nabla, \overline{a} \# x : s \vdash^E M : s'}{\nabla \vdash^E \lambda^{\overline{a}} x : s.\, M : s^{\overline{a}} \Rightarrow s'}$$

$$(\mathsf{AP})\quad \frac{\nabla \vdash^E F : s^{\overline{a}} \Rightarrow s' \qquad \nabla \vdash^E \overline{a} \# A : s}{\nabla \vdash^E F\ A : s'}$$

$$(\mathsf{AE})\quad \frac{\nabla^{\#\overline{a}} \vdash^E M : s}{\nabla \vdash^E M : s}\ (\overline{a} \# (\nabla, M)) \qquad (\mathsf{WEAK})\quad \frac{\nabla \vdash^E M : s}{\nabla' \vdash^E M : s}\ (\nabla \leq \nabla')$$

$$(\mathsf{SUB})\quad \frac{\nabla' \vdash^E \overline{a_i} \# N_i : s_i \qquad \nabla \vdash^E M : s'}{\nabla' \vdash^E M\{N_1, \ldots, N_n / x_1, \ldots, x_n\} : s'}$$

In rule (SUB) $\quad \nabla \stackrel{\text{def}}{=} \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$ and $1 \leq i \leq n$

Table 3
NLC Typing Rules for a Given Th

- We define an **expression-in-context** as a judgement of the form $\nabla \vdash^E M : s$ where ∇ is a freshness environment, M is an α-equivalence class of NLC-terms (an expression) and s is a type.
- An **equation-in-context** is a judgement of the form $\nabla \vdash^E M \approx M' : s$ where $\nabla \vdash^E M : s$ and $\nabla \vdash^E M' : s$.

A NLC-theory Th is a pair (Sg, Ax), where Sg is a NLC-signature and Ax is a collection of equations-in-context. We shall use Th to inductively define a subset of expressions-in-context and equations-in-context. Any expression-in-context that has a derivation is a **typed expression**; and any such equation-in-context is a **theorem**. The set of typed expressions and theorems of a NLC-theory Th is the least set of judgements containing the axioms of Th and closed under the rules in Table 3 and Table 4. We indicate that any judgement J has a derivation in theory Th by writing $Th \rhd J$.

Remark 3.1 Justified by [6] we use the following abbreviation: for $\nabla \vdash M : s$ and $\overline{a} \subseteq \mathbb{A}$ $(\overline{a} \# s)$, we write $\nabla^{\#\overline{b}} \stackrel{\text{def}}{=} \overline{a}_1 \cup \overline{b} \# x_1 : s_1, ..., \overline{a}_n \cup \overline{b} \# x_n : s_n$ and

$$\nabla \vdash^E \overline{a} \# M : s \stackrel{\text{def}}{=} \nabla^{\#\overline{b}} \vdash^E M \approx (\overline{a}\,\overline{b}) * M : s.$$

In the transposition, $\overline{a} \in \mathbb{A}^n$ is sugar for a tuple of the atoms in the set $\overline{a}$ and

$$(\mathsf{REF})\ \frac{\nabla \vdash^E M : s}{\nabla \vdash^E M \approx M : s} \qquad (\mathsf{SYM})\ \frac{\nabla \vdash^E M \approx M' : s}{\nabla \vdash^E M' \approx M : s}$$

$$(\mathsf{TRANS})\ \frac{\nabla \vdash^E M \approx M' : s \qquad \nabla \vdash^E M' \approx M'' : s}{\nabla \vdash^E M \approx M'' : s}$$

$$(\mathsf{WEAK})\ \frac{\nabla \vdash^E M \approx M' : s}{\nabla' \vdash^E M \approx M' : s}\ (\nabla \leq \nabla')$$

$$(\mathsf{AE})\ \frac{\nabla^{\#\overline{a}} \vdash^E M \approx M' : s}{\nabla \vdash^E M \approx M' : s}\ (\overline{a} \# (\nabla, M, M'))$$

$$(\mathsf{PERM})\ \frac{\nabla \vdash^E M : s}{\nabla^{\#ds(\pi,\pi')} \vdash^E \pi * M \approx \pi' * M : \pi \cdot s}\ (ds(\pi,\pi') \# (\nabla, M))$$

$$(\mathsf{BF})\ \frac{\nabla, \overline{a} \# x : s \vdash^E M : s' \qquad \nabla \vdash^E \overline{a} \# N : s}{\nabla \vdash^E (\lambda^{\overline{a}} x : s.\, M)\ N \approx M\{N/x\} : s'}$$

$$(\mathsf{EF})\ \frac{\nabla \vdash^E M : s^{\overline{a}} \Rightarrow s'}{\nabla \vdash^E \lambda^{\overline{a}} x : s.\, (M\ x) \approx M : s^{\overline{a}} \Rightarrow s'}\ (x \notin fv(M))$$

$$(\mathsf{CF})\ \frac{\nabla, \overline{a} \# x : s \vdash^E M \approx M' : s'}{\nabla \vdash^E \lambda^{\overline{a}} x : s.\, M \approx \lambda^{\overline{a}} x : s.\, M' : s^{\overline{a}} \Rightarrow s'}$$

$$(\mathsf{CA})\ \frac{\nabla \vdash^E \overline{a} \# A_i : s \quad \nabla \vdash^E F_1 \approx F_2 : s^{\overline{a}} \Rightarrow s' \quad \nabla \vdash^E A_1 \approx A_2 : s}{\nabla \vdash^E F_1\ A_1 \approx F_2\ A_2 : s'}\ (\mathrm{i}{=}1{,}2)$$

$$(\mathsf{SUB})\ \frac{\begin{array}{c}\nabla' \vdash^E \overline{a_i} \# N'_i : s_i \\ \nabla' \vdash^E \overline{a_i} \# N_i : s_i \qquad \nabla' \vdash^E N_i \approx N'_i : s_i \qquad \nabla \vdash^E M \approx M' : s'\end{array}}{\nabla' \vdash^E M\{N_1, \ldots, N_n/x_1, \ldots, x_n\} \approx M'\{N'_1, \ldots, N'_n/x_1, \ldots, x_n\} : s'}$$

$ds(\pi, \pi')$ is the **disagreement set**: $\{a \in \mathbb{A} \mid \pi(a) \neq \pi(a)\}$

In rule (SUB) $\quad \nabla \stackrel{\text{def}}{=} \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$ and $1 \leq i \leq n$

Table 4
NLC Equation Rules for a Given Th

$\overline{b} \in \mathbb{A}^n$ is any/some fresh tuple of the same size such that $\overline{b} \# (\nabla, \overline{a}, M)$. If $Th \rhd \nabla \vdash^E \overline{a} \# M : s$ then we may legitimately call the judgement a theorem, but we will usually call it a **freshness assertion**.

The role that the judgements $\nabla \vdash^E \overline{a} \# M : s$ play leads to a crucial difference

between NEL and NLC. Consider the rule AP. Since F has type $s^{\overline{a}} \Rightarrow s'$ then $\overline{a}$ must be fresh for the argument A, formally encoded as $\nabla \vdash^E \overline{a} \# A : s$. Thus the type system rules have equations-in-context as hypotheses, and the equation rules have expressions-in-context as hypotheses. Thus theorems and typed expressions are mutually inductively defined. Obviously this complicates our proofs, at least in comparison to NEL, and leads to some subtleties which we explain in due course.

We have two more lemmas that are crucial for proving some important facts about NLC. Lemma 3.2 is used in induction steps in which a binding variable in an abstraction also occurs in the environment (of the abstraction): For an example induction see the proof on page 11 of Lemma 3.3, and [28] (page 169) for a detailed explanation of the problem. Lemma 3.3 is used in proving Proposition 3.4; the proposition underpins our semantics and classifying category construction.

Lemma 3.2 (Variable Equivariance of Judgements) *All typed expressions, and all theorems (hence freshness assertions too), are equivariant under variable swapping. More precisely, for any two distinct variables x, y, and where $(x\,y) \bullet -$ denotes variable swapping (see page 7), we have*

$$Th \rhd \nabla \vdash^E M : s \Longrightarrow Th \rhd (x\,y) \bullet \nabla \vdash^E (x\,y) \bullet M : s$$
$$Th \rhd \nabla \vdash^E M \approx M' : s \Longrightarrow Th \rhd (x\,y) \bullet \nabla \vdash^E (x\,y) \bullet M \approx (x\,y) \bullet M' : s$$

Lemma 3.3 *$Th \rhd \nabla, \overline{a} \# x : s \vdash^E M : s'$ if and only if $Th \rhd \nabla, \pi \cdot \overline{a} \# x : \pi \cdot s \vdash^E M\{\pi^{-1}x/x\} : s'$ and similarly for equations.*

Proof. Since permutations are isomorphisms we only need to prove one direction of the implication. We have to prove, by (mutual) induction over the rules in Table 3 and 4,

$$(\forall Th \rhd \nabla' \vdash^E [M]_\alpha : s') \quad [$$
$$(\forall\, \nabla, \overline{a}, \pi, x, s) \quad (\nabla' \equiv \nabla, \overline{a} \# x : s$$
$$\Longrightarrow Th \rhd \nabla, \pi \cdot \overline{a} \# x : \pi \cdot s \vdash^E [M\{\pi^{-1}x/x\}]_\alpha : s')) \quad]$$

$$(\forall Th \rhd \nabla' \vdash^E [M]_\alpha \approx [M']_\alpha : s') \quad [$$
$$(\forall\, \nabla, \overline{a}, \pi, x, s) \quad (\nabla' \equiv \nabla, \overline{a} \# x : s$$
$$\Longrightarrow Th \rhd \nabla, \pi \cdot \overline{a} \# x : \pi \cdot s \vdash^E [M\{\pi^{-1}x/x\}]_\alpha \approx [M'\{\pi^{-1}x/x\}]_\alpha : s')) \quad]$$

In the remainder of this example proof we concentrate only on illustrating the care we take over dealing with proofs involving capture avoiding re-naming.

Rule (ABS) : Consider the following instance

$$\frac{\nabla', \overline{b} \# y : t \vdash^E [N]_\alpha : t'}{\nabla' \vdash^E [\lambda^{\overline{b}} \boxed{\ y\ } : t.\, N]_\alpha : t^{\overline{b}} \Rightarrow t'}\ \text{ABS}$$

As an illustration of the proof, consider an (arbitrary) instantiation of $(\forall\ \nabla, \overline{a}, \pi, x, s)$ such that $\nabla' \equiv \nabla, \overline{a} \# y : s$ and $y \equiv x$. For Induction Property Closure, since $[(\lambda^{\overline{b}} y : t.\, N)\{\pi^{-1}y/y\}]_\alpha = [\lambda^{\overline{b}} y : t.\, N]_\alpha$, we have to prove that

$$\nabla, \pi \cdot \overline{a} \# y : \pi \cdot s \vdash^E [\lambda^{\overline{b}} y : t.\, N]_\alpha : t' \quad (\diamond)$$

We cannot immediately invert ABS since the binding $\boxed{y}$ occurs in ∇'. Choosing distinct y' we have $[\lambda^{\overline{b}} y : t.\, N]_\alpha = [\lambda^{\overline{b}} y' : t.\, (y'\ y) \bullet N]_\alpha \quad (\dagger)$ so we may now invert ABS to get

$$\nabla, \overline{a} \# y : s, \overline{b} \# y' : t \vdash^E [(y'\ y) \bullet N]_\alpha : t'$$

and hence by the variable equivariance of judgements, Lemma 3.2,

$$\nabla, \overline{a} \# y' : s, \overline{b} \# y : t \vdash^E [N]_\alpha : t'$$

Therefore by induction with $(\forall\ \nabla, \overline{a}, \pi, x, s)$ locally instantiated to $\nabla, \overline{b} \# y : t, \overline{a}, \pi, y', s$ we have

$$\nabla, \pi \cdot \overline{a} \# y' : \pi \cdot s, \overline{b} \# y : t \vdash^E [N\{\pi^{-1}y'/y'\}]_\alpha = [N]_\alpha : t'$$

since $y' \notin var(N)$. Hence by Lemma 3.2 we have

$$\nabla, \pi \cdot \overline{a} \# y : \pi \cdot s, \overline{b} \# y' : t \vdash^E [(y'\ y) \bullet N]_\alpha : t'$$

and $(\diamond)$ follows from this using an instance of ABS, and $(\dagger)$. □

In order to define our categorical semantics, we will require Proposition 3.4 and Proposition 3.5.

Proposition 3.4 ($*$ preserves Typed Expressions and "Equalities") *Given a theory Th,*

$$Th \rhd \nabla \vdash^E M : s \text{ implies } Th \rhd \nabla \vdash^E \pi * M : \pi \cdot s$$
$$Th \rhd \nabla \vdash^E M \approx M' : s \text{ implies } Th \rhd \nabla \vdash^E \pi * M \approx \pi * M' : \pi \cdot s$$

Proposition 3.5 (Atom Equivariance of Judgements) *Given a theory Th,*

$$Th \rhd \nabla \vdash^E M : s \text{ implies } Th \rhd \pi \cdot \nabla \vdash^E \pi \cdot M : \pi \cdot s$$
$$Th \rhd \nabla \vdash^E M \approx M' : s \text{ implies } Th \rhd \pi \cdot \nabla \vdash^E \pi \cdot M \approx \pi \cdot M' : \pi \cdot s$$

4 A Sound Categorical Semantics

FM-Cartesian Closed Categories. Underlying intuition for FM-cccs starts by considering internal categories $\mathcal{I}$ in *FMNom*. Such structures, while necessary for modelling NLC, are not sufficiently rich: to give meaning to NLC terms we must encode permutation actions as morphisms—an additional requirement on $\mathcal{I}$. We

follow the "type (i) approach": axiomatising $\mathcal{I}$ externally and equipping with permutation morphisms, yields *a category with finitely supported internal permutation actions*. We then obtain good notions of products and exponentials by stipulating coherence conditions between these structures and the internal permutation action; these are *cartesian closed perm-categories*. The (additional, external) axiomatisation of freshness properties yields *FM-cccs*. Further details of FM-categories are in [7].

A category $\mathcal{C}$ has an **internal permutation action** if for each $\pi \in Perm$ and $C \in ob\,\mathcal{C}$ there is a $\mathcal{C}$-arrow $\pi_C : C \to \pi \cdot C$ such that ι_C is the identity id_C and $(\pi' \circ \pi)_C = \pi'_{\pi \cdot C} \circ \pi_C$, where $\pi \cdot C$ is defined to be the codomain of π_C. An internal permutation action is **finitely supported** if every arrow $f : C \to D$ in $\mathcal{C}$ is finitely supported with respect to the permutation action $\pi \cdot f \stackrel{\text{def}}{=} \pi_D \circ f \circ (\pi^{-1})_{\pi \cdot C}$. We call a category with a finitely supported permutation action a **perm-category**. A perm-category has **equivariant products** if it has finite products, and the internal permutation action preserves the projections (hence also preserves the product objects). A perm-category with equivariant finite products has **equivariant exponentials** if it is cartesian closed and the internal permutation action preserves the evaluation morphism $\pi \cdot ev_{A,B} = ev_{\pi \cdot A, \pi \cdot B}$ (and hence exponential objects are preserved). A perm-category with equivariant finite products has **fresh inclusions** if for every finite set of atoms $\overline{a} \subseteq \mathbb{A}$ and $\mathcal{C}$-object C such that $\overline{a} \# C$ we have a $\mathcal{C}$-arrow $i^{\overline{a}}_C : C^{\#\overline{a}} \to C$ for which the following properties hold:

(i) (Equivariance): $\pi \cdot i^{\overline{a}}_C = i^{\pi \cdot \overline{a}}_{\pi \cdot C}$;

(ii) (Sets of Atoms): $i^{\emptyset}_C = id_C$ and $i^{\overline{a}}_C \circ i^{\overline{a'}}_{C^{\#\overline{a}}} = i^{\overline{a} \cup \overline{a'}}_C$;

(iii) (Products): $i^{\overline{a}}_{C_1 \times C_2} = i^{\overline{a}}_{C_1} \times i^{\overline{a}}_{C_2}$;

(iv) (Internal permutation action): if $supp(\pi) \# C$ then $\pi_{C^{\#supp(\pi)}}$ is equal to the identity $id_{C^{\#supp(\pi)}}$;

(v) (Epi When Fresh): If we have parallel $\mathcal{C}$-arrows $f, g : C \to D$ such that $f \circ i^{\overline{a}}_C = g \circ i^{\overline{a}}_C$ and $\overline{a} \# (f, g)$, then $f = g$;

(vi) (Freshness): Let $f : C \to D$ be such that $\overline{a} \# D$. Define $\dagger(f, \overline{a}) \stackrel{\text{def}}{=} (\exists\, \overline{b})(\overline{b} \# (\overline{a}, f) \wedge (\overline{a}\,\overline{b})_D \circ f \circ i^{\overline{b}}_C = f \circ i^{\overline{b}}_C)$. If $\dagger(f, \overline{a})$ holds then there is a unique $f^* : C \to D^{\#\overline{a}}$, the **image restriction** of f, such that $i^{\overline{a}}_D \circ f^* = f$.

A perm-category with equivariant finite products and fresh inclusions is an **FM-category** and if it also has equivariant exponentials we call it an **FM-ccc**. The category *FMSet* of FM-sets is an FM-ccc, with the (equivariant) exponential of FM-sets X and Y being the FM-set $X \Rightarrow_{fs} Y$ of finitely supported functions from X to Y, and with $i^{\overline{a}}_X : X^{\#\overline{a}} \stackrel{\text{def}}{=} \{x \in X \mid \overline{a} \# x\} \hookrightarrow X$ as fresh inclusions. FM-cpos are another example.

We will use the functor $(-) \Rightarrow (+) : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$, which is defined by $(A, B) \mapsto A \Rightarrow B$ and $(f, g) \mapsto f \Rightarrow g \stackrel{\text{def}}{=} \lambda(g \circ ev \circ (id_{A \Rightarrow B} \times f))$. An auxiliary lemma is used in establishing that our semantics is sound; the proof is routine category theory. Its use is illustrated briefly on page 16.

Lemma 4.1

(i) *For any* $f : A \times B \to C$ *we have* $\pi \cdot \lambda(f) = \lambda(\pi \cdot f)$

(ii) $\pi_B \circ ev_{A,B} = ev_{\pi\cdot A,\pi\cdot B} \circ (\pi_{A\Rightarrow B} \times \pi_A)$.

(iii) $\pi \cdot (f \Rightarrow g) = \pi \cdot f \Rightarrow \pi \cdot g$

(iv) $\pi_{A\Rightarrow B} = \pi^{-1}_{\pi\cdot A} \Rightarrow \pi_B$

(v) *For any* $f : A \times B \to C$ *we have* $\pi_{B\Rightarrow C} \circ \lambda(f) = \lambda(\pi_C \circ f \circ (id \times \pi^{-1}_{\pi\cdot B}))$

(vi) *For any* $f : A \times B \to C$ *and* $g : A' \to A$, $\lambda(f) \circ g = \lambda(f \circ (g \times id_B))$

Remark 4.2 Each freshness property has a simple intuition. We give one example for (Freshness). Let $f : X \to Y$ be finitely supported in *FMSet*, $x \in X$ and $a \# Y$. By choosing $b \# a, f$ *and* $b \# x$ we have $(f \circ i^b_C)(x) = f(x)$ and the condition $\dagger(f, a)$ amounts to $(b \# a, f) \wedge (a\,b) \cdot f(x) = f(x)$. But since $b \# x$ we can also deduce $b \# f(x)$, so we have $(b \# a, f(x)) \wedge (a\,b) \cdot f(x) = f(x)$. Hence $f(x) \in Y^{\#a}$ and so f image restricts (with $f^* : x \mapsto f(x)$).

A Sound Categorical Semantics. We wish to define a categorical semantics which will interpret typed expressions $Th \rhd \nabla \vdash^E M : s$ as morphisms $[\![\nabla \vdash^E M : s]\!] : [\![\nabla]\!] \longrightarrow [\![s]\!]$ in an FM-ccc $\mathcal{C}$. However we have seen that NLC is dependently typed: in particular the type system and equation system are mutually inductively defined. This means that we cannot give a simple recursive definition of a function $[\![-]\!]$ over (well-typed) expressions [31,33]. However, we can give such a definition of a partial semantic function, which is defined only when certain equations are themselves satisfied by $[\![-]\!]$.

We also deal with a further complication. See rule AP which has hypothesis $\nabla \vdash^E \overline{a} \# A : s$. We wish to define, following Remark 4.2, the semantics of $\nabla \vdash^E \overline{a} \# A : s$ as $[\![\nabla \vdash^E \overline{a} \# A : s]\!] \stackrel{\text{def}}{=} [\![\nabla \vdash^E A : s]\!]^*$—but this morphism is defined only if the condition $\dagger([\![\nabla \vdash^E A : s]\!], \overline{a})$ holds! Thus we also need to factor this requirement into our semantics and soundness theorem.

We can now define the semantics. Let $\mathcal{C}$ be a FM-ccc and Sg a NLC-signature. Then a ***Sg*-structure** $\mathcal{M}$ in $\mathcal{C}$ is specified by giving:

- An equivariant map $[\![-]\!] : Gnd_{Sg} \longrightarrow ob\,\mathcal{C}$. We extend to the map $[\![-]\!] : Type_{Sg} \longrightarrow ob\,\mathcal{C}$ via structural recursion ($[\![s^{\overline{a}} \Rightarrow s']\!] \stackrel{\text{def}}{=} [\![s]\!]^{\#\overline{a}} \Rightarrow [\![s']\!]$) and this is easily seen to be equivariant too, since $\mathcal{C}$ has equivariant structure.
- An equivariant map $[\![-]\!] : Fun_{Sg} \longrightarrow ob\,\mathcal{C}$ where for each higher order function constant $c : s$ we have $[\![c]\!] : 1 \longrightarrow [\![s]\!]$ (recall that $\mathcal{C}$ has finite products—hence an equivariant terminal object).

Let $\nabla = \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n \in Env_{Sg}$ be a freshness environment. Then we define the $\mathcal{C}$-object $[\![\nabla]\!]$ by $[\![\nabla]\!] \stackrel{\text{def}}{=} [\![s_1]\!]^{\#\overline{a}_1} \times ... \times [\![s_n]\!]^{\#\overline{a}_n}$. We define a notion of satisfaction for both expressions-in-context and equations-in-context. Let $\mathcal{M}$ be a structure for a NLC-signature in an FM-ccc $\mathcal{C}$ and consider the binary relation $\blacktriangleright$ in Table 5. Table 5 specifies a partial function $J \mapsto [\![J]\!]$ from judgements to morphisms $[\![J]\!]$ in $\mathcal{C}$. Given $\nabla \vdash^E M : s$ or $\nabla \vdash^E \overline{a} \# M : s$ we say that $\mathcal{M}$ **satisfies**

$$[\![\nabla, \overline{a_i} \# x_i : s_i \vdash \pi x_i : \pi \cdot s_i]\!] \blacktriangleright \pi_{[\![s_i]\!]} \circ i^{\overline{a_i}}_{[\![s_i]\!]} \circ pr_i : [\![\nabla, \overline{a_i} \# x_i : s_i]\!] \longrightarrow [\![s_i]\!]^{\#\overline{a_i}} \longrightarrow [\![s_i]\!] \longrightarrow \pi \cdot [\![s_i]\!]$$

$$[\![\nabla \vdash c : s]\!] \blacktriangleright [\![c]\!] \circ ! : [\![\nabla]\!] \to 1 \to [\![s]\!]$$

$$\frac{[\![\nabla, \overline{a} \# x : s \vdash M : s']\!] \blacktriangleright m : [\![\nabla]\!] \times [\![s]\!]^{\overline{a}} \to [\![s']\!]}{[\![\nabla \vdash \lambda^{\overline{a}} x : s.M : s^{\overline{a}} \Rightarrow s']\!] \blacktriangleright \lambda(m) : [\![\nabla]\!] \to ([\![s]\!]^{\#\overline{a}} \Rightarrow [\![s']\!])}$$

$$\frac{[\![\nabla \vdash F : s^{\overline{a}} \Rightarrow s']\!] \blacktriangleright f : [\![\nabla]\!] \to ([\![s]\!]^{\#\overline{a}} \Rightarrow [\![s']\!]) \quad [\![\nabla \vdash \overline{a} \# A : s]\!] \blacktriangleright \theta : [\![\nabla]\!] \to [\![s]\!]^{\#\overline{a}}}{[\![\nabla \vdash F\ A : s']\!] \blacktriangleright ev \circ \langle f, \theta \rangle : [\![\nabla]\!] \to ([\![s]\!]^{\#\overline{a}} \Rightarrow [\![s']\!]) \times [\![s]\!]^{\#\overline{a}} \to [\![s']\!]}$$

$$\frac{[\![\nabla \vdash^E M : s]\!] \blacktriangleright m}{[\![\nabla \vdash^E \overline{a} \# M : s]\!] \blacktriangleright m^*} \quad \dagger(m, \overline{a})$$

Table 5
Semantics of Higher Order Functions

the judgement if the morphism $[\![\nabla \vdash^E M : s]\!] : [\![\nabla]\!] \longrightarrow [\![s]\!]$ or $[\![\nabla \vdash^E \overline{a} \# M : s]\!] : [\![\nabla]\!] \longrightarrow [\![s]\!]^{\#\overline{a}}$ in $\mathcal{C}$ is defined (that is, the partial function $J \mapsto [\![J]\!]$ is defined). If so we write $[\![\nabla \vdash^E M : s]\!]\Downarrow$ or $[\![\nabla \vdash^E \overline{a} \# M : s]\!]\Downarrow$. Generally, $[\![J]\!]\Downarrow \stackrel{\text{def}}{=} (\exists j)([\![J]\!] \blacktriangleright j)$. We may write $[\![J]\!]$ or even $[\![J]\!]\Downarrow$ for morphism j. Given $\nabla \vdash^E M \approx M' : s$ we say that $\mathcal{M}$ **satisfies** it if both $[\![\nabla \vdash^E M : s]\!]\Downarrow$ and $[\![\nabla \vdash^E M' : s]\!]\Downarrow$ and they are equal morphisms in $\mathcal{C}$. We say that $\mathcal{M}$ is a **model** of a NLC theory $Th = (Sg, Ax)$ if $\mathcal{M}$ satisfies all of the equations-in-context in Ax. With this, we have our soundness theorem:

Theorem 4.3 (Soundness) *Let Th be a* NLC *theory and $\mathcal{M}$ a model of Th in an FM-ccc. Then every typed expression $Th \rhd \nabla \vdash^E M : s$, freshness assertion $Th \rhd \nabla \vdash^E \overline{a} \# M : s$ and theorem $Th \rhd \nabla \vdash^E M \approx M' : s$ is satisfied by $\mathcal{M}$.*

We need the following intermediate results to prove the soundness theorem. We adopt a direct approach to proving that our semantics is compositional with respect to substitution, which reduces some overhead from the approach in [7]. Note that we appeal to Propositions 3.4 and Proposition 3.5 to ensure that the NLC judgements mentioned below are properly defined. We shall write $L \asymp R$ to mean that $L\Downarrow \iff R\Downarrow$ and that $L = R$.

Lemma 4.4 (Semantic Id, Inclusion, Int. Perm. Action, Projection) *Given a freshness environment $\nabla = \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$ then we have*

(i) $id_{[\![\nabla]\!]} \asymp \langle [\![\nabla \vdash \overline{a_1} \# x_1 : s_1]\!], ..., [\![\nabla \vdash \overline{a_n} \# x_n : s_n]\!] \rangle$

(ii) $i^{\overline{a}}_{[\![\nabla]\!]} \asymp \langle [\![\nabla^{\#\overline{a}} \vdash \overline{a_1} \# x_1 : s_1]\!], ..., [\![\nabla^{\#\overline{a}} \vdash \overline{a_n} \# x_n : s_n]\!] \rangle$

(iii) $\pi_{[\![\nabla]\!]} \asymp \langle [\![\nabla \vdash \pi \cdot \overline{a_1} \# \pi x_1 : \pi \cdot s_1]\!], ..., [\![\nabla \vdash \pi \cdot \overline{a_n} \# \pi x_n : \pi \cdot s_n]\!] \rangle$

(iv) $pr_{[\![\nabla_j]\!]} : [\![\nabla_1]\!] \times [\![\nabla_2]\!] \to [\![\nabla_j]\!] \asymp \langle [\![\boldsymbol{\nabla_1 \cup \nabla_2 \vdash \overline{a_i} \# x_i : s_i}]\!] \rangle$, *where $\nabla_1, \nabla_2 \in Env_{Sg}$ have disjoint domains but are such that $\nabla_j = \nabla$ for $j = 1$ and 2.*

Lemma 4.5 (Useful Semantic Factorisations "$[\![\xi]\!] = [\![\xi]\!] \circ m$")

(i) The function $[\![-]\!] : Env_{Sg} \to Env_{Sg}$ is equivariant.

(ii) $[\![\pi \cdot \nabla \vdash^E \pi \cdot M : \pi \cdot s]\!] \asymp \pi \cdot [\![\nabla \vdash^E M : s]\!]$

(iii) $[\![\nabla \vdash \pi * M : \pi \cdot s]\!] \asymp \pi_{[\![s]\!]} \circ [\![\nabla \vdash M : s]\!]$

(iv) $$[\![\nabla, \pi\cdot\overline{a} \# x : \pi\cdot s \vdash^E M\{\pi^{-1}/x\} : s']\!] \asymp [\![\nabla, \overline{a} \# x : s \vdash^E M : s']\!] \circ (id_{[\![\nabla]\!]} \times \pi^{-1}_{[\![\pi\cdot s]\!]^{\#\pi\cdot\overline{a}}})$$

(v) *Given* $\nabla \leq \nabla'$ *there exists an arrow* $weak : [\![\nabla']\!] \to [\![\nabla]\!]$ *such that for any typed expression* $\nabla \vdash^E M : s$, $[\![\nabla' \vdash^E M : s]\!] \asymp [\![\nabla \vdash^E M : s]\!] \circ weak$.

(vi) $[\![\nabla^{\#\overline{a}} \vdash^E M : s]\!] \asymp [\![\nabla \vdash^E M : s]\!] \circ i^{\overline{a}}$ *where* $\overline{a} \# \nabla$.

Proof. We illustrate proofs of Lemma 4.5 part i and ii:

(i) Following the definitions in our paper together with the properties of a perm-category, we have

$$\begin{aligned}
\pi\cdot[\![\nabla]\!] &= \pi\cdot([\![s_1]\!]^{\#\overline{a_1}} \times ... \times [\![s_n]\!]^{\#\overline{a_n}}) \\
&= (\pi\cdot[\![s_1]\!]^{\#\overline{a_1}} \times ... \times \pi\cdot[\![s_n]\!]^{\#\overline{a_n}}) \\
&= ((\pi\cdot[\![s_1]\!])^{\#\pi\cdot\overline{a_1}} \times (\pi\cdot[\![s_n]\!])^{\#\pi\cdot\overline{a_n}}) \\
&= ([\![\pi\cdot s_1]\!]^{\#\pi\cdot\overline{a_1}} \times [\![\pi\cdot s_n]\!]^{\#\pi\cdot\overline{a_n}}) \\
&= [\![[\pi\cdot\overline{a_1} \# x_1 : \pi\cdot s_1, ..., \pi\cdot\overline{a_n} \# x_n : \pi\cdot s_n]]\!] \\
&= [\![\pi\cdot\nabla]\!]
\end{aligned}$$

(ii) Proof by induction on the structure of M

$$(\forall M) \quad [\quad (\forall\, \nabla, \pi, s) \quad (\pi\cdot[\![\nabla \vdash^E M : s]\!] \asymp [\![\pi\cdot\nabla \vdash^E \pi\cdot M : \pi\cdot s]\!])) \quad]$$

SUSP: It directly follows from the categorical semantics that

$$[\![\pi\cdot\nabla, \pi\cdot\overline{a} \# x : \pi\cdot s \vdash^E \pi\cdot\pi' x : \pi\cdot\pi'\cdot s]\!]\Downarrow$$

and $[\![\nabla, \overline{a} \# x : s \vdash^E \pi' x : \pi'\cdot s]\!]\Downarrow$

The equality follows by basic properties of FM-cccs.

CONST: It is immediate that $[\![\nabla \vdash^E c : s]\!]\Downarrow$ and $[\![\pi\cdot\nabla \vdash^E \pi\cdot c : \pi\cdot s]\!]\Downarrow$. The equality follows from the fact that $[\![-]\!] : Fun_\Sigma \to ob\,\mathcal{C}$ is equivariant.

LAM-ABS: Suppose $[\![\pi\cdot\nabla \vdash^E \pi\cdot(\lambda^{\overline{a}} x : s.\, M) : \pi\cdot(s^{\overline{a}} \Rightarrow s')]\!]\Downarrow$ and it is equal to f_π. By the definition of the meta-level permutation action and the inductively defined semantics we get

$$[\![\pi\cdot\nabla \vdash^E \lambda^{\pi\cdot\overline{a}} x : \pi\cdot s.\, \pi\cdot M : (\pi\cdot s)^{\pi\cdot\overline{a}} \Rightarrow \pi\cdot s']\!] \blacktriangleright \lambda(m_\pi)$$

for some m_π where $[\![\pi\cdot\nabla, \pi\cdot\overline{a} \# x : \pi\cdot s \vdash^E \pi\cdot M : \pi\cdot s']\!] \blacktriangleright m_\pi$. By induction we deduce that $[\![\nabla, \overline{a} \# x : s \vdash^E M : s']\!] \blacktriangleright m$ such that $\pi\cdot m = m_\pi$. We then apply the rule for semantics of abstraction to obtain $[\![\nabla \vdash^E \lambda^{\overline{a}} x : s.\, M : s^{\overline{a}} \Rightarrow s']\!] \blacktriangleright \lambda(m)$, that is, $[\![\nabla \vdash^E \lambda^{\overline{a}} x : s.\, M : s^{\overline{a}} \Rightarrow s']\!]\Downarrow$. The definitional existence proof in the converse direction follows by similar reasoning. We now need to show that $f_\pi = \pi\cdot\lambda(m)$.

$$\begin{aligned} f_\pi &\stackrel{\text{def}}{=} \lambda(m_\pi) \\ &= \lambda(\pi \cdot m) && \text{(induction)} \\ &= \pi \cdot \lambda(m) && \text{(Lemma 4.1 (i))} \end{aligned}$$

APP: Suppose $[\![\pi \cdot \nabla \vdash^E \pi \cdot (F\ A) : \pi \cdot s']\!]\Downarrow$ and it is equal to t_π. By the definition of the meta-level permutation action and the rule for semantics of applications

$$[\![\pi \cdot \nabla \vdash^E (\pi \cdot F)\ (\pi \cdot A) : \pi \cdot s']\!] \blacktriangleright ev \circ \langle f_\pi, \theta_\pi \rangle$$

for $[\![\pi{\cdot}\nabla \vdash^E \pi{\cdot}F : (\pi \cdot s)^{\pi\cdot\overline{a}} \Rightarrow \pi{\cdot}s']\!] \blacktriangleright f_\pi$ and $[\![\pi{\cdot}\nabla \vdash^E \pi{\cdot}\overline{a} \mathbin{\#} \pi{\cdot}A : \pi{\cdot}s]\!] \blacktriangleright \theta_\pi$. We have $[\![\pi \cdot \nabla \vdash^E \pi \cdot \overline{a} \mathbin{\#} \pi \cdot A : \pi \cdot s]\!] \blacktriangleright {\alpha_\pi}^*$ by the rule for freshness assertion semantics where $[\![\pi \cdot \nabla \vdash^E \pi \cdot A : \pi \cdot s]\!] \blacktriangleright \alpha_\pi$ such that $\dagger(\pi \cdot \overline{a}, \alpha_\pi)$. Given that $\blacktriangleright$ is a partial function, we have that $\theta_\pi = {\alpha_\pi}^*$. By induction we get $[\![\nabla \vdash^E F : s^{\overline{a}} \Rightarrow s']\!] \blacktriangleright f$ and $[\![\nabla \vdash^E A : s]\!] \blacktriangleright \alpha$ such that $f_\pi = \pi \cdot f$ and $\alpha_\pi = \pi \cdot \alpha$. We now deduce from $\dagger(\pi \cdot \overline{a}, \alpha_\pi)$ that $\dagger(\overline{a}, \alpha)$ holds: Let $\overline{a'} \mathbin{\#} (\overline{a}, \alpha)$. It follows immediately that $\pi \cdot \overline{a'} \mathbin{\#} (\pi \cdot \overline{a}, \pi \cdot \alpha)$ and hence from $\dagger(\pi \cdot \overline{a}, \pi \cdot \alpha)$ we obtain equation (1). In the equations below, we write internal permutation actions τ_C as τ_- since the source-target data does not play a significant role in our reasoning, and indeed is probably obfuscating:

$$(\pi \cdot \overline{a'}\,\pi \cdot \overline{a})_- \circ (\pi \cdot \alpha) \circ i = (\pi \cdot \alpha) \circ i \tag{1}$$

$$(\pi \cdot \overline{a'}\,\pi \cdot \overline{a})_- \circ \pi_- \circ \alpha \circ \pi_-^{-1} \circ i = \pi_- \circ \alpha \circ \pi_-^{-1} \circ i \tag{2}$$

$$\pi_- \circ (\overline{a'}\,\overline{a})_- \circ \alpha \circ \pi_-^{-1} \circ i = \pi_- \circ \alpha \circ \pi_-^{-1} \circ i \tag{3}$$

$$(\overline{a'}\,\overline{a})_- \circ \alpha \circ i \circ \pi_-^{-1} = \alpha \circ i \circ \pi_-^{-1} \tag{4}$$

$$(\overline{a'}\,\overline{a})_- \circ \alpha \circ i = \alpha \circ i \tag{5}$$

By definition of the FM-ccc permutation action on morphisms we obtain equation (2). The transposition notation $(\overline{a'}\,\overline{a})$ is short for $(a'_1\,a_1){\circ}\ldots{\circ}(a'_k\,a_k)$. Since in $Perm$, $\pi{\circ}(c\,d) = (\pi(c)\,\pi(d)){\circ}\pi$ holds generally for single transpositions $(c\,d)$, and since permutation actions satisfy $(\tau' \circ \tau)_C = \tau'_{\tau \cdot C} \circ \pi_C$ we have

$$\pi_- \circ (\overline{a'}\,\overline{a})_- = (\pi \circ (\overline{a'}\,\overline{a}))_- = ((\pi \cdot \overline{a'}\,\pi \cdot \overline{a}) \circ \pi)_- = (\pi \cdot \overline{a'}\,\pi \cdot \overline{a})_- \circ \pi_-$$

This gives us equation (3). Any internal permutation action $(\tau_C : C \to \tau \cdot C \mid C \in ob\,\mathcal{C})$ is a natural transformation $Id \to \tau \cdot -$ and in particular so is π_-^{-1}. Since also π_- is iso, equation (4) holds. Finally since π_-^{-1} is iso we obtain (5). Hence, $\dagger(\overline{a}, \alpha)$ holds.

We can now apply the rule for freshness assertion semantics to obtain $[\![\nabla \vdash^E \overline{a} \mathbin{\#} A : s]\!] \blacktriangleright \alpha^*$, followed by the rule for application semantics to get $[\![\nabla \vdash^E F\ A : s']\!] \blacktriangleright ev \circ \langle f, \alpha^* \rangle$. Hence, we have $[\![\nabla \vdash^E F\ A : s']\!]\Downarrow$. The definitional existence proof in the converse direction follows by similar reasoning.

We now show that $t_\pi = \pi \cdot (ev \circ \langle f, \alpha^* \rangle)$. Note that $(\pi \cdot \alpha)^* = \pi \cdot \alpha^* \quad (\Phi)$ holds: This follows immediately from the universal property of inclusion image restriction and the definition of $\pi \cdot -$. Hence

$$\begin{aligned}
t_\pi &\stackrel{\text{def}}{=} ev_{([\![\pi\cdot s]\!]^{\#\pi\cdot\overline{a}},[\![\pi\cdot s']\!])} \circ \langle f_\pi, \theta_\pi \rangle \\
&= ev_{([\![\pi\cdot s]\!]^{\#\pi\cdot\overline{a}},[\![\pi\cdot s']\!])} \circ \langle f_\pi, {\alpha_\pi}^* \rangle && (\theta_\pi = {\alpha_\pi}^*) \\
&= ev_{(\pi\cdot([\![s]\!]^{\#\overline{a}}),\pi\cdot[\![s']\!])} \circ \langle \pi \cdot f, (\pi \cdot \alpha)^* \rangle && \text{(induction)} \\
&= ev_{(\pi\cdot([\![s]\!]^{\#\overline{a}}),\pi\cdot[\![s']\!])} \circ \langle \pi \cdot f, \pi \cdot \alpha^* \rangle && (\Phi) \\
&= ev_{(\pi\cdot([\![s]\!]^{\#\overline{a}}),\pi\cdot[\![s']\!])} \circ (\pi \cdot \langle f, \alpha^* \rangle) && \text{(equivariant products)} \\
&= (\pi \cdot ev_{([\![s]\!]^{\#\overline{a}},[\![s']\!])}) \circ (\pi \cdot \langle f, \alpha^* \rangle) && \text{(equivariant exponentials)} \\
&= \pi \cdot (ev_{([\![s]\!]^{\#\overline{a}},[\![s']\!])} \circ \langle f, \alpha^* \rangle) && \text{(equivariance of } \circ\text{)}
\end{aligned}$$

□

Proposition 4.6 (Compositional Semantics) *Let* $\nabla \stackrel{\text{def}}{=} \overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n$. *Suppose, for theory Th, we have the typed expression* $\nabla \vdash^E M : s$ *and freshness assertions* $\nabla' \vdash^E \overline{a}_i \# N_i : s_i$ *for each* i. *Then we have* $\nabla' \vdash^E M\{\boldsymbol{N_i}/\boldsymbol{x_i}\} : s$. *Moreover, if* $[\![\nabla \vdash^E M : s]\!]\Downarrow$ *and* $[\![\nabla' \vdash^E \overline{a}_i \# N_i : s_i]\!]\Downarrow$ *for each* i *then we have* $[\![\nabla' \vdash^E M\{\boldsymbol{N_i}/\boldsymbol{x_i}\} : s]\!]\Downarrow$ *and further* $[\![\nabla' \vdash^E M\{\boldsymbol{N_i}/\boldsymbol{x_i}\} : s]\!] = [\![\nabla \vdash^E M : s]\!] \circ \langle [\![\boldsymbol{\nabla' \vdash^E \overline{a}_i \# N_i : s_i}]\!] \rangle$.

The proofs of Lemmas 4.4 and 4.5 require a combination of direct calculations and inductions over the structure of terms. Note that the proof of Proposition 4.6 is by induction over the structure of M and does not require a complicated statement that is provable by mutual induction. The intuition is that, as one would expect, the semantics of expressions is derivation independent. We are now in a position to prove Theorem 4.3.

Proof. This proof does proceed by a mutual induction establishing the satisfaction of all judgement forms. Induction Property Closure for all the rules in Table 3 and Table 4 is similar to our example:

(AP): We need to show that $[\![\nabla \vdash^E F\ A : s']\!]\Downarrow \quad (\diamond)$. By induction we have $[\![\nabla \vdash^E F : s^{\overline{a}} \Rightarrow s']\!] \blacktriangleright f \quad (1)$. Recalling that satisfaction of the freshness assertion is the satisfaction of an equation

$$\nabla \vdash^E \overline{a} \# A : s \quad \stackrel{\text{def}}{=} \quad (\forall \,/\, \exists\, \overline{a'} \# (\nabla, \overline{a}, A)) \quad (\nabla^{\#\overline{a'}} \vdash^E A \approx (\overline{a}\,\overline{a'}) * A : s)$$

we have $[\![\nabla^{\#\overline{a'}} \vdash^E A : s]\!] \blacktriangleright \theta$ and $[\![\nabla^{\#\overline{a'}} \vdash^E (\overline{a}\,\overline{a'}) * A : s]\!] \blacktriangleright \theta'$ with $\theta = \theta'$. Hence by Lemma 4.4 vi we have $\theta = \alpha \circ i$ where $[\![\nabla \vdash^E A : s]\!] \blacktriangleright \alpha$ and by Lemma 4.4 iii and 4.4 vi we have $\theta' = (\overline{a}\,\overline{a'})_{[\![s]\!]} \circ \alpha \circ i$. From the (Epi When Fresh) property of FM-cccs we have $\alpha = (\overline{a}\,\overline{a'})_{[\![s]\!]} \circ \alpha$, that is $\dagger(\alpha, \overline{a})$. Hence $[\![\nabla \vdash^E \overline{a} \# A : s]\!] \blacktriangleright \alpha^* \quad (2)$. From (1) and (2) we have $(\diamond)$, with definition $ev \circ \langle f, \alpha^* \rangle$.

Property Closure for the rules in Table 4 is trivial for (REF) (SYM) (TRANS). (WEAK) uses Lemma 4.4 v. (AE) uses Lemma 4.4 vi and Lemma 4.4 ii. (PERM) uses Lemma 4.4 iii and Lemma 4.4 vi. (BF) is quite similar to the details given for (AP).□

5 A Complete Categorical Semantics

In order to obtain a completeness result we need a way to construct a cartesian closed category out of NLC syntax. To do this we augment the types, expressions and rules with a form of dependently typed atom-abstraction. In doing so we arrive at the final form of NLC (with abstraction) for which we have a categorical model that is both sound and complete. Please note that we only give a summary of the details in this preliminary paper; a substantial journal version will follow.

We augment the type system with types of the form $[a]s$. We augment the collection of terms with abstraction and concretion denoted by $\langle a\rangle\, M$ and $M \mathbin{@} a$ respectively, and by a form of local scoping $\mathsf{fr}\, a.M$.

Occurrences of a in $\langle a\rangle\, M$ are not bound. The permutation actions on the resulting terms (and expressions) are defined in the expected way. The type system and equation system appears in Table 6 on page 20.

The equations specify forms of beta and eta equality, ensure that term forming operations are congruences, and that the $\langle a\rangle\, M$ abstraction operator on expressions is equated with $\langle a'\rangle\, M'$ provided that the two expressions given by swapping out the a and a' for a fresh atom b are provable equal in the logic (so "binding" is encoded at the level of formal equations). As for semantics, in *FMSet* one should think of the usual semantic notions of abstraction and concretion modelling $\langle a\rangle\, M$ and $M \mathbin{@} a$, and the expression $\mathsf{fr}\, a.M$ as the syntactic analogue of $\mathsf{fresh}\ a\ \mathsf{in}\ F(a)$ (see [30], the Freshness Theorem).

We also need a richer categorical structure to achieve completeness. For any FM-category $\mathcal{C}$, there is a family of categories $(\mathcal{C}^{\#a} \mid a \in \mathbb{A})$ where $ob\,\mathcal{C}^{\#a}$ consists of those $C \in ob\,\mathcal{C}$ for which $a \# C$. Given such $C, C' \in ob\,\mathcal{C}^{\#a}$, then $f : C \to C' \in mor\,\mathcal{C}$ is a morphism in $\mathcal{C}^{\#a}$ just in case $a \# f$. The basic properties of fresh inclusions ensure that each $\mathcal{C}^{\#a}$ is indeed a category, and moreover that there is a functor $(-)^{\#a} : \mathcal{C}^{\#a} \to \mathcal{C}$. For the remainder of this section we fix on an atom a that specifies the functor $(-)^{\#a}$. We shall require this functor to have a right adjoint $[a](-)$ (moreover, an equivalence) and for there to be a family of morphisms $conc_b : ([a]C)^{\#b} \to (a\,b)\cdot C$. These structures are required to satisfy the commutativity properties which are needed in order to soundly model the equations (see Table 6). For example, for every $D \in ob\,\mathcal{C}^{\#a}$, $X \in ob\,\mathcal{C}$, and $a', b \# X$, where $\eta_{a,D}$ is the counit of the adjunction, we have

$$\begin{array}{ccccc} D^{\#a} & \xrightarrow{m} & (a\,a')\cdot X & \xrightarrow{(a'\,b)_{(a\,a')\cdot X}} & (a\,b)\cdot X \\ {\scriptstyle i}\downarrow & & & & \uparrow{\scriptstyle conc_b} \\ D & & \xrightarrow[F^*]{} & & ([a]X)^{\#b} \end{array}$$

with F being the morphism

$$D \xrightarrow{\eta_{a,D}} [a]D^{\#a} \xrightarrow{[a]m} [a](a\,a')\cdot X \xrightarrow{[a](a\,a')_{(a\,a')\cdot X}} [a]X$$

$$(\mathsf{AABS})\quad \frac{\nabla^{\#a} \vdash^E M : (a\,a') \cdot s}{\nabla \vdash^E \langle a' \rangle\, M : [a]s}\ (a \# \nabla, M, a' \# s)$$

$$(\mathsf{CONC})\quad \frac{\nabla \vdash^E b \# F : [a]s}{\nabla^{\#a} \vdash^E F \mathbin{@} b : (a\,b) \cdot s}\ (a \# \nabla)[b = a \vee b \# s]$$

$$(\mathsf{LN})\quad \frac{\nabla^{\#a} \vdash^E a \# M : s}{\nabla \vdash^E \mathsf{fr}\, a.M : s}\ (a \# \nabla)[a \# s]$$

$$(\mathsf{BAA})\quad \frac{\nabla^{\#a} \vdash^E M : (a\,a') \cdot s \qquad \nabla \vdash^E b \# \langle a' \rangle\, M : [a]s}{\nabla^{\#a} \vdash^E (\langle a' \rangle\, M) \mathbin{@} b \approx (a'\,b) \cdot M : (a\,b) \cdot s}\ \big(a \# \nabla, M, a' \# s\big)$$

$$(\mathsf{EAA})\quad \frac{\nabla \vdash^E b \# F : [a]s}{\nabla \vdash^E \langle b \rangle\, (F \mathbin{@} b) \approx F : [a]s}\ (a \# \nabla, b \# F)$$

$$(\mathsf{BINDAA})\quad \frac{\nabla^{\#a,b} \vdash^E (b\,a') \cdot M \approx (b\,a'') \cdot M' : (a\,a') \cdot s}{\nabla \vdash^E \langle a' \rangle\, M \approx \langle a'' \rangle\, M' : [a]s}\ \begin{array}{l}(a \# \nabla, M, M', a', a'' \# s,\\ b \# a, a', a'', M, M', s)\end{array}$$

$$(\mathsf{CC})\quad \frac{\nabla \vdash^E b \# F : [a]s \qquad \nabla \vdash^E F \approx F' : [a]s}{\nabla^{\#a} \vdash^E F \mathbin{@} b \approx F' \mathbin{@} b : (a\,b) \cdot s}\ (a \# \nabla)[a = b \vee b \# s]$$

$$(\mathsf{LNFr})\quad \frac{\nabla^{\#a} \vdash^E a \# M : s}{\nabla^{\#a} \vdash^E \mathsf{fr}\, a.M \approx M : s}\ (a \# \nabla, a \# M)[a \# s]$$

$$(\mathsf{LNS})\quad \frac{\nabla^{\#\overline{b}} \vdash^E \overline{b} \# M : s}{\nabla^{\#\overline{b} \setminus \{a,a'\}} \vdash^E \mathsf{fr}\, a.\mathsf{fr}\, a'.M \approx \mathsf{fr}\, a'.\mathsf{fr}\, a.M : s}\ (\overline{b} \# \nabla,\ a, a' \in \overline{b})[\overline{b} \# s]$$

$$(\mathsf{LNFS})\quad \frac{\nabla^{\#a'}, \overline{a} \# x : s \vdash^E M : s'}{\nabla \vdash^E \mathsf{fr}\, a'.\lambda^{\overline{a}} x : s.\, M \approx \lambda^{\overline{a}} x : s.\, \mathsf{fr}\, a'.M : s^{\overline{a}} \Rightarrow s'}\ (a' \notin \overline{a})$$

Table 6
NLC Augmented Typing and Equation Rules for a Given Th

Further

$$\begin{array}{ccc} D & \xrightarrow{\quad F \quad} & [a]X \\ {\scriptstyle \eta_{a,D}}\downarrow & & \| \\ [a]D^{\#a} & \xrightarrow[[a]((a\,b)_{(a\,b)\cdot X} \circ conc_a \circ F^* \circ i^a_D)]{} & [a]X \end{array}$$

where

$$D^{\#a} \xrightarrow{i^a_D} D \xrightarrow{F^*} ([a]X)^{\#b} \xrightarrow{conc_b} (a\,b) \cdot X \xrightarrow{(a\,b)_{(a\,b)\cdot X}} X$$

$$\frac{[\![\nabla^{\#a} \vdash^E M : (a\,a') \cdot s]\!] \blacktriangleright m : [\![\nabla]\!]^{\#a} \to (a\,a') \cdot [\![s]\!]}{[\![\nabla \vdash^E \langle a' \rangle M : [a]s]\!] \blacktriangleright [a]((a\,a')_{(a\,a')\cdot [\![s]\!]} \circ m) \circ \eta_{a,[\![\nabla]\!]} : [\![\nabla]\!] \to [a][\![\nabla]\!]^{\#a} \to [a][\![s]\!]}$$

$$\frac{[\![\nabla \vdash^E F : [a]s]\!] \blacktriangleright f : [\![\nabla]\!] \to [a][\![s]\!]}{[\![\nabla^{\#a} \vdash^E F @ b : (a\,b) \cdot s]\!] \blacktriangleright conc_b \circ f^* \circ i^a_{[\![\nabla]\!]} : [\![\nabla]\!]^{\#a} \to [\![\nabla]\!] \to ([a][\![s]\!])^{\#b} \to (a\,b) \cdot [\![s]\!]} \quad \dagger(f,b)$$

$$\frac{[\![\nabla^{\#b} \vdash^E b \# M : s]\!] \blacktriangleright \theta : [\![\nabla]\!]^{\#b} \to [\![s]\!]^{\#b}}{[\![\nabla \vdash^E \mathsf{fr}\, b.M : s]\!] \blacktriangleright \eta^{-1}_{b,[\![\nabla]\!]} \circ [b]\theta \circ \eta_{b,[\![s]\!]} : [\![\nabla]\!] \to [b][\![\nabla]\!]^{\#b} \to [b][\![s]\!]^{\#b} \to [\![s]\!]}$$

Table 7
Semantics of Abstraction and Concretion

Suppose that we also require the adjoints to commute. We call such FM-cccs with this additional structure ИFM-cccs; it is these categories that yield a sound and complete semantics for NLC. The semantics of abstraction and concretion appears in Table 7.

An example of such a category is *FMSet*. The action of the functor $(-)^{\#a}$ sends any FM-function $f : X \to Y \in FMSet^{\#a}$ to $f^{\#a} : X^{\#a} \to Y^{\#a}$ where $f^{\#a}(x \in X^{\#a}) \stackrel{\text{def}}{=} f(x) \in Y^{\#a}$ is easily seen to be well-defined. The action of the right adjoint $[a](-)$ is defined on $f : X \to Y$ by setting

$$[a]X \stackrel{\text{def}}{=} \{\langle a' \rangle\, x \mid a' \# X \wedge x \in (a\,a') \cdot X\}$$

where $\langle a' \rangle\, x$ is the abstraction operator of Gabbay and Pitts [14], and further $[a]f(a' \in [a]X) \stackrel{\text{def}}{=} \mathsf{fresh}\ b\ \mathsf{in}\ \langle b \rangle\, ((a\,b) \cdot f)(a' @ b)$. The verification that we have an adjunction satisfying the stated properties is a rather length calculation which we omit from this paper.

The Classifying Category and Categorical Completeness. The notion of classifying category, topos, etc is a standard one in category theory [10,22]. To prove completeness we now show that we can build an FM-ccc from the syntax of a NLC theory (Proposition 5.1), together with a generic model [10] (Propositions 5.3 and 5.4).

Proposition 5.1 (Classifying Category) *For every* NLC*-theory* Th *we can define a classifying FM-ccc* $Cl(Th)$ *which is built from the syntax of* Th*. An object is a freshness environment* $\nabla \stackrel{\text{def}}{=} (\overline{a}_1 \# x_1 : s_1, ..., \overline{a}_n \# x_n : s_n)$*. If* $\nabla' \stackrel{\text{def}}{=} (\overline{a'}_1 \# x'_1 : s'_1, ..., \overline{a'}_m \# x'_m : s'_m)$ *then a morphism* $\delta \stackrel{\text{def}}{=} ([M_1]_\approx, \ldots, [M_m]_\approx) : \nabla \to \nabla'$ *is a list of typed expressions such that for* $1 \leq i \leq m$ *we have* $Th \rhd \nabla \vdash^E \overline{a}'_i \# M_i : s'_i$*, and* $[M_i]_\approx$ *is the equivalence class of those* T *such that* $Th \rhd \nabla \vdash^E M_i \approx T : s'_i$*.*

Remark 5.2 We explain, with a simple example, how we are able to construct exponentials in the classifier. Consider $(a_1 \# x_1 : s_1) \Rightarrow (a'_1 \# x'_1 : s'_1)$. One would imagine that, whatever the exponential is, it should somehow involve the type ${s_1}^{a_1} \Rightarrow {s'_1}^{a'_1}$ which is not legitimate in NLC. However, consider the following, recalling that in an ИFM-ccc the adjoints commute

$$\mathcal{C}(C_1^{\#a_1}, {C'_1}^{\#a'_1}) \cong \mathcal{C}^{\#a'_1}([a'_1]C_1^{\#a_1}, C'_1) \cong \mathcal{C}^{\#a'_1}(([a'_1]C_1)^{\#a_1}, C'_1)$$

We can mimic the above isomorphisms in the syntax of NLC in order to construct exponentials, and in fact we can show that

$$(a_1 \# x_1 : s_1) \Rightarrow (a'_1 \# x'_1 : s'_1) \stackrel{\text{def}}{=} a'_1 \# f : \underbrace{([a'_1]s_1)^{\#a_1} \Rightarrow s'_1}_{\text{NLC } type}$$

and that this easily extends to the general case of $\nabla_1 \Rightarrow \nabla_2$.

Proposition 5.3 *The* ***generic*** *Sg-structure $\mathcal{G}$ of $Th = (Sg, Ax)$ in $Cl(Th)$ is given by defining $[\![\gamma]\!]_{\mathcal{G}} =_{def} (\emptyset \# x : s)$ where γ is any ground type from Sg. If c is a constant with typing $c : s$ then $[\![c]\!]_{\mathcal{G}} \stackrel{\text{def}}{=} ([c]_{\approx}) : 1 \stackrel{\text{def}}{=} () \longrightarrow (\emptyset \# x : s)$ is well defined since $Th \rhd [] \vdash^E c : s$ Further, suppose that $Th \rhd \nabla \vdash^E M : s$. Then $[\![\nabla \vdash^E M : s]\!]_{\mathcal{G}} \blacktriangleright [M]_{\approx} : \nabla \to (\emptyset \# v : s)$*

Proposition 5.4 *The generic structure $\mathcal{G}$ is a model of any $Th = (Sg, Ax)$.*

Theorem 5.5 (Completeness) *The categorical semantics of* NLC*-theories in FM-cccs is complete: Let Th be a* NLC*-theory. If any equation-in-context for Th is satisfied in all FM-ccc models of Th, then it is a theorem.*

6 Category Theory/Type Theory Correspondence

Clouston [7] demonstrated a categorical type theory correspondence between NEL and FM-categories; we have established a similar correspondence between NLC and FM-cccs. Recall [10] that the correspondence result for standard λ-calculus and cartesian closed categories is slightly more restricted than the one for EL and categories with finite products: Due to the covariant nature of exponentials, components of homomorphisms of models must be restricted to isomorphisms.

Theorem 6.1 *The category $Cl(Th)$ is a* ***classifying category*** *for* NLC*-theories in the sense that for every model $\mathcal{M}$ of Th in a ИFM-ccc $\mathcal{D}$ there is a unique ИFM-ccc functor $F_{\mathcal{M}} : Cl(Th) \to \mathcal{D}$ such that $F_{\mathcal{M}}$ composes with the generic model to yield $\mathcal{M}$.*

Now take a definition of homomorphism $h : \mathcal{M} \to \mathcal{N}$ of models of an NLC-theory Th in an ИFM-ccc $\mathcal{C}$ consisting of **equivariant** isomorphisms $h_\gamma : [\![\gamma]\!]_{\mathcal{M}} \cong [\![\gamma]\!]_{\mathcal{M}}$, where $h_{s^{\overline{a}} \Rightarrow s'}$ is given by $(h_s^{\#\overline{a}})^{-1} \Rightarrow h_{s'}$ (and $\overline{a} \# s$ ensures homomorphisms are well defined). For a NLC-theory Th and a small ИFM-ccc $\mathcal{C}$, the **category of models** $\mathcal{M}od_{\cong}(Th, \mathcal{C})$ consists of the Th models and homomorphisms. An ИFM-ccc functor $F : \mathcal{C} \to \mathcal{D}$ is an ИFM-functor that preserves exponentials and commutes with the adjunction. We can define $FMccc_{\cong}(\mathcal{C}, \mathcal{D})$ as a category with ИFM-ccc functors as objects and finitely supported natural isomorphisms as morphisms.

Theorem 6.2 *We have $FMccc_{\cong}(Cl(Th), \mathcal{D}) \simeq \mathcal{M}od_{\cong}(Th, \mathcal{D})$ for any* NLC*-theory Th and ИFM-ccc $\mathcal{D}$. For any ИFM-ccc $\mathcal{C}$, we have $Cl(Th(\mathcal{C})) \simeq \mathcal{C}$. For any* NLC*-theory Th we have $Th \simeq Th(Cl(Th))$.*

7 Solutions, Open Questions, and Further Work

Exploiting Atom-Dependent Types. Clouston [4] observes that name-abstraction and concretion in *FMSet* cannot be captured by a NEL theory. This is related to the fact that concretion is a partial function, which can only be applied to arguments that meet certain freshness conditions. In the total concretion theory in Section 8 of [4] (page 15; MFPS 2010), Clouston describes concretion functions of the form $con_a : Name.s \to s$ where $Name.s$ is the name-abstraction type. Now $con_a\ x$ is well-formed only if $a \# x$. NEL does not support such partiality. But in NLC we have exploited the new dependent type system to yield a solution.

Internal and 2-Categorical Approaches Could this paper be simplified by considering internalisation in one of the FM-toposes? We cannot give a definitive answer: a deep investigation must wait for future work, but here are a few observations. Consider even the basic notion of perm-category. A perm-category is internal to *FMNom*; but *an internal FMNom-category is not a perm-category* since the morphism permutation π_C is not directly captured by the internalisation. So it is not clear to us that the notion of FM-ccc could be extracted by internalisation. Going further, it is also not clear how the atom-set-partiality of our higher types can be (usefully) captured. However, even if it can, this misses a central point of our paper: a direct investigation into the interplay of higher order types and the freshness relation via a domain specific formal type theory. Possibly if one sought a direct route to "some kind of" completeness result an internal approach might work, but we are trying to do more than that. What is true is that the "nominal" world still needs to be better understood from a "2-categorical" viewpoint, and there are a number of open questions.

Future Work. Recall our motivation for this work: to develop a formal framework for nominal higher order functions, with a view to proving it a conservative extension over NEL by nominal gluing. Nominal gluing remains work in progress, but our preliminary results about the Yoneda Lemma and cartesian closure of nominal functor categories appear in [12]. From such a gluing proof, we might be able to extract a form of categorical normalisation result, taking the work in a more applied direction through the construction of some form of abstract machine for NLC along with an implementation. Is there some form of nominal categorical abstract machine?

From the Computer Science perspective, we have taken great care in specifying NLC formally and care with proofs that involve quite subtle intricacies arising from α-equivalence in the nominal setting, and the (variable) equivariance of judgements and rules. We have attempted to avoid the traps (explained in [28]) that others have fallen into. As such, it would be an interesting project to study a mechanisation of NLC.

How much further can one take categorical correspondences for nominal logics/type theories? We are considering product and coproduct types, and of course one might study computational monad types, numbers, and more [20]. Going still further there is the general consideration of Martin Löf dependent type theory

[27,32], nominal and FM analogues, and corresponding categorical structures. We are also investigating Henkin style models as have Gabbay and Mulligan [17]. Cheney [3] has studied the properties of a type theory that mixes functions, and atoms as first class citizens, along with a name abstraction operator. While discussing others' work, it is interesting to note that *type dependency* is a common feature of studies involving computational type-and-effect systems. Examples are [34,1].

We have considered the possibility that the original NEL could be presented using dependent types in place of freshness assertions. However, the resulting type theory might be different. Such dependently typed theories, in which $\overline{a} \# x : s$ is wholly replaced by $x : s^{\overline{a}}$, could be more expressive than NEL theories. This remains future work.

Acknowledgement

We thank Ranald Clouston for very detailed comments; John Power for being helpful and generous with his time over a public holiday; and for other useful comments and observations from Martin Hyland and Bill Lawvere. We must also thank others who have provided useful thoughts that we hope have improved this paper. Finally, and most importantly, we thank Andrew Pitts for extensive discussions about an earlier version of this paper which contained an error and assisting with its correction.

References

[1] Nick Benton, Andrew Kennedy, Lennart Beringer, and Martin Hofmann. Relational Semantics for Effect-Based Program Transformations with Dynamic Allocation. In *Proc. of the 9th ACM SIGPLAN international conference on Principles and Practice of Declarative Programming*, PPDP '07, pages 87–96, New York, NY, USA, 2007. ACM.

[2] Clemens Berger, Paul-Andre Mellies, and Mark Weber. Monads with Arities and their Associated Theories. *Journal of Pure and Applied Algebra*, 216(89):2029–2048, 2012.

[3] James Cheney. A Dependent Nominal Type Theory. *Logical Methods in Computer Science*, 8(1), 2012.

[4] Ranald Clouston. Binding in Nominal Equational Logic. *Electr. Notes Theor. Comput. Sci.*, 265:259–276, 2010.

[5] Ranald Clouston. Nominal Lawvere Theories. In *WoLLIC'11*, pages 67–83, 2011.

[6] Ranald Clouston. Nominal Logic with Equations Only. In *Logical Frameworks, Metalanguages and Theory of Programming*, pages 44–57, 2011.

[7] Ranald Clouston. Nominal Lawvere Theories: A Category Theoretic Account of Equational Theories with Names. *Journal of Computer and System Sciences*, 2013.

[8] Ranald Clouston and Andrew M. Pitts. Nominal Equational Logic. *Electron. Notes Theor. Comput. Sci.*, 172:223–257, 2007.

[9] R. L. Crole. On Fixpoint Objects and Gluing Constructions. *Applied Categorical Structures*, 4(2 & 3):251–281, 1996. This volume is a Special Edition for the European Colloquium on Category Theory, Tours, France.

[10] Roy L. Crole. *Categories for Types*. Cambridge University Press, 1993.

[11] Roy L. Crole. α-Equivalence Equalities. *Theoretical Computer Science*, 433:1–19, May 2012.

[12] Roy L. Crole and Frank Nebel. The Yoneda Lemma and Cartesian Closure in the FM-World. Submitted, 2013.

[13] P.J. Freyd and A. Scedrov. *Categories, Allegories*. Elsevier Science Publishers, 1990. Appears as Volume 39 of the North-Holland Mathematical Library.

[14] Murdoch Gabbay and Andrew M. Pitts. A New Approach to Abstract Syntax with Variable Binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

[15] Murdoch J. Gabbay. Foundations of Nominal Techniques: Logic and Semantics of Variables in Abstract Syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.

[16] Murdoch J. Gabbay and Aad Mathijssen. Nominal Universal Algebra: Equational Logic with Names and Binding. *Journal of Logic and Computation*, 19(6):1455–1508, December 2009.

[17] Murdoch James Gabbay and Dominic P. Mulligan. Nominal Henkin Semantics: Simply-Typed Lambda-Calculus Models in Nominal Sets. In *LFMTP*, pages 58–75, 2011.

[18] J.R. Hindley and J.P. Seldin. *Introduction to Combinators and the Lambda Calculus*, volume 1 of *London Mathematical Society Student Texts*. Cambridge University Press, 1988.

[19] Martin Hyland and John Power. The Category Theoretic Understanding of Universal Algebra: Lawvere Theories and Monads. *Electr. Notes Theor. Comput. Sci.*, 172:437–458, 2007.

[20] Neelakantan R. Krishnaswami and Nick Benton. Adding Equations to System F Types. In *ESOP*, pages 417–435, 2012.

[21] J. Lambek. From λ-calculus to cartesian closed categories. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[22] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

[23] F.W. Lawvere. *Functorial Semantics of Algebraic Theories*. PhD thesis, Columbia University, 1963. Summary appears in Proc. of the National Academy of Science, 50:869–873, 1963.

[24] E. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. SV, 1976.

[25] James McKinna and Robert Pollack. Some Lambda Calculus and Type Theory Formalized. *JAR*, 1998.

[26] Paul-André Melliès. Segal Condition Meets Computational Effects. In *LICS*, pages 150–159, 2010.

[27] B. Nordström, K. Petersson, and J.M. Smith. *Programming in Martin-Löf's Type Theory*, volume 7 of *Monographs on Computer Science*. Oxford University Press, 1990.

[28] A. M. Pitts. Nominal Logic, A First Order Theory of Names and Binding. *Information and Computation*, 186:165–193, 2003.

[29] A. M. Pitts. Structural Recursion with Locally Scoped Names. *Journal of Functional Programming*, 21(3):235–286, 2011.

[30] A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*, volume 57 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2013.

[31] Andrew M. Pitts. Categorical Logic. In *Handbook of Logic in Computer Science: Volume 5: Logic and Algebraic Methods*, pages 39–123, Oxford, UK, 2000. Oxford University Press.

[32] Thomas Streicher. Independence Results for Calculi of Dependent Types. In *Category Theory and Computer Science*, pages 141–154, 1989.

[33] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness, and Independence Results*, volume XII of *Progress in Theoretical Computer Science*. Basel: Birkhäuser Verlag, 1991.

[34] Jacob Thamsborg and Lars Birkedal. A Kripke Logical Relation for Effect-Based Program Transformations. In *Proc of the 16th ACM SIGPLAN international conference on functional programming*, ICFP '11, pages 445–456, New York, NY, USA, 2011. ACM.