N. Kobayashi and B. C. Pierce (Eds): Fourth International Symposium on Theoretical Aspects of Computer Software (TACS2001), LNCS Vol. 2215, pp. 219–242, 2001.

# Nominal Logic: A First Order Theory of Names and Binding [⋆]

Andrew M. Pitts

University of Cambridge, Computer Laboratory, Cambridge CB3 0FD, UK
Andrew.Pitts@cl.cam.ac.uk

**Abstract.** This paper formalises within first-order logic some common practices in computer science to do with representing and reasoning about syntactical structures involving named bound variables (as opposed to nameless terms, explicit substitutions, or higher order abstract syntax). It introduces *Nominal Logic*, a version of first-order many-sorted logic with equality containing primitives for renaming via name-swapping and for freshness of names, from which a notion of binding can be derived. Its axioms express key properties of these primitives, which are satisfied by the *FM-sets* model of syntax introduced in [11]. Nominal Logic serves as a vehicle for making two general points. Firstly, name-swapping has much nicer logical properties than more general forms of renaming while at the same time providing a sufficient foundation for a theory of structural induction/recursion for syntax modulo $\alpha$-conversion. Secondly, it is useful for the practice of operational semantics to make explicit the *equivariance property* of assertions about syntax—namely that their validity is invariant under name-swapping.

## 1 Introduction

It is commonplace, when using formal languages in computer science or mathematical logic, to abstract away from details of concrete syntax in terms of strings of symbols and instead work solely with parse trees—the 'abstract syntax' of a language. Doing so gives one access to two extremely useful and inter-related tools: definition by recursion on the structure of parse trees and proof by induction on that structure. However, conventional abstract syntax is not abstract enough if the formal language involves variable-binding constructs. In this situation the common practice of human (as opposed to computer) provers is to say one thing and do another. We say that we will quotient the collection of parse trees by a suitable equivalence relation of $\alpha$-conversion, identifying trees up to renaming of bound variables; but then we try to make the use of $\alpha$-equivalence classes as implicit as possible by dealing with them via suitably chosen representatives. How to make such suitable choices of representatives is well understood, so much so that is has a name—the 'Barendregt Variable Convention': choose a representative parse tree whose bound variables are *fresh*, i.e. mutually distinct and distinct from any (free) variables in the current context. This informal

[⋆] Research funded by UK EPSRC grant GR/R07615.

practice of confusing an $\alpha$-equivalence class with a member of the class with sufficiently fresh bound variables has to be accompanied by a certain amount of hygiene on the part of human provers: our constructions and proofs have to be independent of which particular fresh names we choose for bound variables. Nearly always, the verification of such independence properties is omitted, because it is tedious and detracts from more interesting business at hand. Of course this introduces a certain amount of informality into 'pencil-and-paper' proofs that cannot be ignored if one is in the business of producing fully formalised, machine-checked proofs. But even if you are not in that business and are content with your pencil and paper, I think there is a good reason to examine this informal use of 'sufficiently fresh names' and put it on a more precise, mathematical footing.

The reason I have in mind has to do with those intuitive and useful tools mentioned above: structural recursion for defining functions on parse trees and structural induction for proving properties of them. Although it is often said that the Barendregt Variable Convention allows one to work with $\alpha$-equivalence classes of parse trees as though they were just parse trees, this is not literally the case when it comes to structural recursion/induction. For example, when dealing with an induction step for a variable-binding construct, it often happens that the step can be proved if the bound variable is chosen sufficiently fresh, but not for an arbitrary bound variable as the induction principle demands. The Barendregt Variable Convention papers over the crack in the proof at this point (by preventing one considering the case of an arbitrary bound variable rather than a fresh one), but the crack is still there. Although one can easily side-step the problem by using a suitable size function on parse trees to replace structural induction with mathematical induction, this is not a very satisfying solution. The size function will be defined by structural recursion and the crucial fact that $\alpha$-equivalent parse trees have the same size will be proved by structural induction, so we are using structural recursion/induction anyway, but somehow not in the direct way we would like. We can do better than this.

Indeed, the work reported in [10, 11, 24] does so, by providing a mathematical notion of 'sufficiently fresh name' that remains very close to the informal practice described above while enabling $\alpha$-equivalence classes of parse trees to gain useful inductive/recursive properties. The theory stems from the somewhat surprising observation that all of the concepts we need ($\alpha$-conversion, freshness, variable-binding, ...) can be defined purely in terms of the operation of *swapping* pairs of names. In particular, the freshness of a name for an object is expressed in terms of the name not being in some finite set of names that *supports* the object—in the sense that swapping any pair of names not in that finite set leaves the object unchanged. This notion of support is weak second order, since it involves an existential quantification over finite sets of names. However, much of the development in [11] only makes use of certain first-order properties of the freshness (i.e. 'not-in-the-support-of') predicate in combination with the swapping operation. This paper presents this first-order theory of names, swapping and freshness, called *Nominal Logic.*

## 2 Equivariant Predicates

The fundamental assumption underlying Nominal Logic is that *the only predicates we ever deal with* (when describing properties of syntax) *are equivariant ones, in the sense that their validity is invariant under swapping* (i.e. transposing, or interchanging) *names.*

Names of what? Names of entities that may be subject to binding by some of the syntactical constructions under consideration. In Nominal Logic these sorts of names, the ones that may be subjected to swapping, will be called *atoms*—the terminology refers back to the origins of the theory in the Fraenkel-Mostowski permutation model of set theory. Atoms turn out to have quite different logical properties from *constants* (in the usual sense of first-order logic) which, being constant, are not subjected to swapping. Note that this distinction between atom and constant has to do with the issue of binding, rather than substitution. A syntactic category of *variables*, by which is usually meant entities that may be subject to substitution, might be represented in Nominal Logic by atoms or by constants, depending upon circumstances: constants will do if we are in a situation where variables are never bound, but can be substituted; otherwise we should use atoms. The interesting point is that we can make this (useful!) distinction between 'bindable' names and names of constants entirely in terms of properties of swapping names, prior to any discussion of substitution and its properties.

Why the emphasis on *swapping* two names, rather than on the apparently more primitive notion of *renaming* one name by another? The answer has to do with the fact that swapping is an idempotent operation: a swap followed by the same swap is equivalent to doing nothing. This means that the class of equivariant predicates, i.e. those whose validity is invariant under name-swapping, has excellent logical properties; it contains the equality predicate and is closed under negation, conjunction, disjunction, existential and universal quantification, formation of least and greatest fixed points of monotone operators, etc., etc. The same is not true for renaming. (For example, the validity of a negated equality between atoms is not necessarily preserved under renaming.)

That we should take equivariance into account when we reason about syntactical structures is one of the main messages of this paper. Even if you do not care about the details of Nominal Logic to be given below, it is worth taking note of the fact that *name swapping and the equivariance property provide a simple and useful foundation for discussing properties of names and binding in syntax.* Here is a simple example to illustrate this point, taken from type theory.

**Example 2.1.** McKinna and Pollack [19] note that in the naïve approach to named bound variables, there is a difficulty with proving the weakening property of type systems by rule induction. For example, consider the usual typing relation assigning simple types to terms of the untyped $\lambda$-calculus. We take the latter to mean $\alpha$-equivalence classes $[t]_\alpha$ of parse trees $t$ given by the grammar

$$t ::= a \mid \lambda a.t \mid t\,t \tag{1}$$

where $a$ ranges over an infinite set of variables. The typing relation takes the form $\varGamma \vdash [t]_\alpha : \tau$ where types $\tau$ are given by the grammar $\tau ::= X \mid \tau \to \tau$ (with $X$ ranging over an infinite collection of type variables) and the typing context $\varGamma$ is a finite partial function from variables to types. The typing relation is inductively generated by axioms and rules following the structure of the parse tree $t$. (If the reader is not familiar with these rules, see [13, Chapter 2], for example; but note that as mentioned in the Introduction, the literature usually does not bother to make a notational distinction between $t$ and $[t]_\alpha$.)

When trying to prove the weakening property of the typing relation

$$(\forall \varGamma)(\forall t)(\forall \tau)\ \varGamma \vdash [t]_\alpha : \tau \ \Rightarrow\ (\forall \tau')(\forall a' \notin \mathrm{dom}(\varGamma))\ \varGamma, a' : \tau' \vdash [t]_\alpha : \tau \tag{2}$$

it is natural to try to proceed by 'rule induction' and show that the predicate $\varphi(\varGamma, [t]_\alpha, \tau)$ given by

$$(\forall \tau')(\forall a' \notin \mathrm{dom}(\varGamma))\ \varGamma, a' : \tau' \vdash [t]_\alpha : \tau$$

defines a relation that is closed under the axioms and rules inductively defining the typing relation and hence contains that relation. But the induction step for the rule for typing $\lambda$-abstractions

$$\frac{\begin{array}{l}\varGamma, a : \tau_1 \vdash [t]_\alpha : \tau_2\\ a \notin \mathrm{dom}(\varGamma)\end{array}}{\varGamma \vdash [\lambda a.t]_\alpha : \tau_1 \to \tau_2} \tag{3}$$

is problematic: we have to prove

$$\varphi(\varGamma, a : \tau_1,\ [t]_\alpha,\ \tau_2) \wedge a \notin \mathrm{dom}(\varGamma)\ \Rightarrow\ \varphi(\varGamma,\ [\lambda a.t]_\alpha,\ \tau_1 \to \tau_2);$$

i.e. given

$$\varphi(\varGamma, a : \tau_1,\ [t]_\alpha,\ \tau_2) \tag{4}$$

and

$$a \notin \mathrm{dom}(\varGamma), \tag{5}$$

we have to prove that

$$\varGamma, a' : \tau' \vdash [\lambda a.t]_\alpha : \tau_1 \to \tau_2 \tag{6}$$

holds for *all* $a' \notin \mathrm{dom}(\varGamma)$ (and all $\tau'$)—and there is a problem with doing this for the case $a' = a$.

But this difficulty with the induction step is easily circumvented if we take equivariance into account. The axioms and rules defining typing are closed under the operations of swapping pairs of variables (and also under swapping pairs of type variables, but we do not need to use that here). For example, if we have an instance of rule (3) and we swap any pair of variables throughout both the hypotheses and the conclusion, we get another valid instance of this rule. It follows from this swapping property of the axioms and rules that the typing relation, being the least relation closed under the axioms and rules, is also closed under the swapping operations. Therefore any assertion about typing that we make by combining the typing relation with other such equivariant predicates (such as '$a \in \mathrm{dom}(\Gamma)$') using the usual logical connectives and quantifiers will be equivariant. In particular the predicate $\varphi$ defined above is equivariant. Thus if we know that (4) holds, then so does $\varphi(\Gamma, a'' : \tau_1,\ [(a''\,a)\cdot t]_\alpha,\ \tau_2)$ for any *fresh* variable $a''$. (Here $(a''\,a)\cdot t$ indicates the parse tree resulting from swapping $a''$ and $a$ throughout $t$.) So by definition of $\varphi$, since $a' \notin \mathrm{dom}(\Gamma, a'' : \tau_1)$, we have $(\Gamma, a'' : \tau_1), a' : \tau' \vdash [(a''\,a)\cdot t]_\alpha : \tau_2$. Since $(\Gamma, a'' : \tau_1), a' : \tau' = (\Gamma, a' : \tau'), a'' : \tau_1$ (we are using partial functions for typing contexts) and $a'' \notin \mathrm{dom}(\Gamma, a' : \tau')$ (by choice of $a''$), we can apply typing rule (3) to conclude that $\Gamma, a' : \tau' \vdash [\lambda a''.((a''\,a)\cdot t)]_\alpha : \tau_1 \to \tau_2$. But $\lambda a''.((a''\,a)\cdot t)$ and $\lambda a.t$ are $\alpha$-equivalent parse trees, so $\Gamma, a' : \tau' \vdash [\lambda a.t]_\alpha : \tau_1 \to \tau_2$ holds. Thus if (4) and (5) hold, so does $\varphi(\Gamma,\ [\lambda a.t]_\alpha,\ \tau_1 \to \tau_2)$ and we have completed the induction step. □

From the considerations of this section we abstract the following ingredients for a language to describe syntax involving names and binding: the language should contain a notion of atom together with operations for swapping atoms in expressions (in general we may need several different sorts of atoms—for example, atoms for variables and atoms for type variables in Example 2.1); and the formulas of the language should all be equivariant with respect to these swapping operations. Atoms and swapping are two of the three novelties of Nominal Logic. The third has to do with the crucial step in the proof in Example 2.1 when we chose a *fresh* variable $a''$: we need to give a freshness relation between atoms and expressions with sufficient properties to make such arguments go through.

## 3 Nominal Logic: Syntax and Semantics

The syntax of Nominal Logic is that of many-sorted first-order logic with equality, augmented by the following extra features.

- The collection of sorts $S$ is partitioned into two kinds

$$\begin{array}{rll} S ::= & A & \textit{sorts of atoms} \\ & D & \textit{sorts of data.} \end{array}$$

- For each sort of atoms $A$ and each sort $S$ there is a distinguished function symbol of arity $A, A, S \longrightarrow S$ whose effect on terms $a : A$, $a' : A$ and $s : S$ we write as the term $(a\,a')\cdot s$ and pronounce '*swap $a$ and $a'$ in $s$*'.
- For each sort of atoms $A$ and each sort $S$ there is a distinguished relation symbol of arity $A, S$ whose effect on terms $a : A$ and $s : S$ we write as the formula $a \mathrel{\#} s$ and pronounce '*$a$ is fresh for $s$*'.

Just as for ordinary first-order logic, a *theory* in Nominal Logic is specified by a *signature* of sort, function and relation symbols, together with a collection of (non-logical) *axioms*, which are first-order formulas involving equality, swapping, freshness and symbols from the signature.

**Example 3.1 ($\lambda$-Terms mod $\alpha$-equivalence, version 1).** Here is an example of a theory in Nominal Logic to which we will return to throughout the paper.

**Sort of atoms:** *Var*
**Sort of data:** *Term*
**Function symbols:**

$$\begin{aligned} \mathit{var} &: \mathit{Var} \longrightarrow \mathit{Term} \\ \mathit{app} &: \mathit{Term}, \mathit{Term} \longrightarrow \mathit{Term} \\ \mathit{lam} &: \mathit{Var}, \mathit{Term} \longrightarrow \mathit{Term} \\ \mathit{subst} &: \mathit{Term}, \mathit{Var}, \mathit{Term} \longrightarrow \mathit{Term} \end{aligned}$$

**Axioms:**

$$(\forall a : \mathit{Var})(\forall t, t' : \mathit{Term})\ \neg\, \mathit{var}(a) = \mathit{app}(t, t') \tag{7}$$

$$(\forall a, a' : \mathit{Var})(\forall t : \mathit{Term})\ \neg\, \mathit{var}(a) = \mathit{lam}(a', t) \tag{8}$$

$$(\forall a : \mathit{Var})(\forall t, t', t'' : \mathit{Term})\ \neg\, \mathit{lam}(a, t) = \mathit{app}(t', t'') \tag{9}$$

$$\begin{aligned} (\forall t : \mathit{Term})\ & (\exists a : \mathit{Var})\ t = \mathit{var}(a) \\ & \vee (\exists t', t'' : \mathit{Term})\ t = \mathit{app}(t', t'') \\ & \vee (\exists a' : \mathit{Var})(\exists t' : \mathit{Term})\ t = \mathit{lam}(a', t') \end{aligned} \tag{10}$$

$$(\forall a, a' : \mathit{Var})\ \mathit{var}(a) = \mathit{var}(a') \Rightarrow a = a' \tag{11}$$

$$(\forall t, t', t'', t''' : \mathit{Term})\ \mathit{app}(t, t') = \mathit{app}(t'', t''') \Rightarrow t = t'' \wedge t' = t''' \tag{12}$$

$$\begin{aligned} (\forall a, a' : \mathit{Var})\, (\forall t, t' : \mathit{Term})\ & \mathit{lam}(a, t) = \mathit{lam}(a', t') \Leftrightarrow \\ & (a = a' \wedge t = t') \vee (a' \mathrel{\#} t \ \wedge\ t' = (a\, a') \cdot t) \end{aligned} \tag{13}$$

$$\begin{aligned} (\forall \vec{x} : \vec{S})\ & (\forall a : \mathit{Var})\ \varphi(\mathit{var}(a), \vec{x}) \\ & \wedge (\forall t, t' : \mathit{Term})\ \varphi(t, \vec{x}) \wedge \varphi(t', \vec{x}) \Rightarrow \varphi(\mathit{app}(t, t'), \vec{x}) \\ & \wedge (\exists a : \mathit{Var})\ a \mathrel{\#} \vec{x} \wedge (\forall t : \mathit{Term})\ \varphi(t, \vec{x}) \Rightarrow \varphi(\mathit{lam}(a, t), \vec{x}) \\ & \Rightarrow (\forall t : \mathit{Term})\ \varphi(t, \vec{x}) \end{aligned} \tag{14}$$

where the free variables of $\varphi$ are in $t, \vec{x}$

$$(\forall t : \mathit{Term})(\forall a : \mathit{Var})\ \mathit{subst}(t, a, \mathit{var}(a)) = t \tag{15}$$

$$(\forall t : \mathit{Term})(\forall a, a' : \mathit{Var})\ \neg\, a = a' \Rightarrow \mathit{subst}(t, a, \mathit{var}(a')) = \mathit{var}(a') \tag{16}$$

$$\begin{aligned} (\forall t, t', t'' : \mathit{Term})\, (\forall a : \mathit{Var})\ & \mathit{subst}(t, a, \mathit{app}(t', t'')) = \\ & \mathit{app}(\mathit{subst}(t, a, t'), \mathit{subst}(t, a, t'')) \end{aligned} \tag{17}$$

$$\begin{aligned} (\forall t, t' : \mathit{Term})\, (\forall a, a' : \mathit{Var})\ & \neg\, a' = a \wedge a' \mathrel{\#} t \Rightarrow \\ & \mathit{subst}(t, a, \mathit{lam}(a', t')) = \mathit{lam}(a', \mathit{subst}(t, a, t')) \end{aligned} \tag{18}$$

Axioms (7)–(10) say that *var*, *app* and *lam* have disjoint images that cover *Term*. Axioms (11)–(13) give the injectivity properties of these three constructors. In particular axiom (13) reflects the fact that equality of terms of sort *Term* should correspond to equality of $\alpha$-equivalence classes of parse trees for the grammar in (1); and freshness $a \mathrel{\#} t$ to the fact that a variable does not occur freely in a parse tree. Axiom (14) formalises a structural induction principle for such $\alpha$-equivalence classes (cf. [11, Theorem 6.8]). Finally, axioms (15)–(18) amount to a structurally recursive definition of capture-avoiding substitution for $\lambda$-terms (cf. [11, Example 6.9]).

This particular theory has a concrete model given by $\alpha$-equivalence classes of parse trees for the grammar in (1). However, as explained in [11], in general the

Nominal Logic notions of atom, swapping and freshness can be given a meaning independent of any particular object-level syntax using *FM-sets*—the Fraenkel-Mostowski permutation model of set theory. The following definitions give a simplified, but essentially equivalent, presentation of FM-sets that emphasises swapping over more general permutations of atoms. At the same time we use a mild generalisation of [11] (mentioned in [10, Sect. 7]) in which the set of atoms is partitioned into countably many different kinds (and we only swap atoms of the same kind).

**Definition 3.2 (Nominal sets).** Fix a countably infinite family $(\mathbb{A}_n \mid n \in \mathbb{N})$ of pairwise disjoint, countably infinite sets. We write $\mathbb{A}$ for the union of all the $\mathbb{A}_n$ and call its elements *atoms*. A *nominal set* $X$ is a set $|X|$ equipped with a well-behaved notion of swapping atoms in elements of the set. By definition this means that for each element $x \in |X|$ and each pair of atoms $a, a'$ of the same kind (i.e. $a, a' \in \mathbb{A}_n$ for some $n \in \mathbb{N}$), we are given an element $(a\,a') \cdot_X x$ of $X$, called the result of swapping $a$ and $a'$ in $x$. These swapping operations are required to have the following properties:

(i) **Equational properties of swapping:** for each $x \in |X|$ and all pairs of atoms of equal sort, $a, a' \in \mathbb{A}_m$ and $b, b' \in \mathbb{A}_n$ (any $m, n \in \mathbb{N}$)

$$(a\,a) \cdot_X x = x \tag{19}$$

$$(a\,a') \cdot_X (a\,a') \cdot_X x = x \tag{20}$$

$$(a\,a') \cdot_X (b\,b') \cdot_X x = ((a\,a')b\ \ (a\,a')b'\ ) \cdot_X (a\,a') \cdot_X x \tag{21}$$

where

$$(a\,a')b \triangleq \begin{cases} a & \text{if } b = a' \\ a' & \text{if } b = a \\ b & \text{otherwise} \end{cases} \tag{22}$$

and similarly for $(a\,a')b'$.

(ii) **Finite support property:** we require that each $x \in |X|$ only involve finitely many atoms, in the sense that given $x$, there exists a finite subset $w \subseteq \mathbb{A}$ with the property that $(a\,a') \cdot_X x = x$ holds for all $a, a' \in \mathbb{A}_n - w$ (any $n \in \mathbb{N}$). It follows that

$$supp_X(x) \triangleq \bigcup_{n \in \mathbb{N}} \{a \in \mathbb{A}_n \mid \{a' \in \mathbb{A}_n \mid (a\,a') \cdot_X x \neq x\} \text{ is not finite}\} \tag{23}$$

is a finite set of atoms (see the proof of [11, Proposition 3.4]), which we call the *support* of $x$ in $X$.

A *morphism of nominal sets*, $f : X \longrightarrow Y$, is by definition a function from the set $|X|$ to the set $|Y|$ that respects the swapping operations in the sense that

$$f((a\,a') \cdot_X x) = (a\,a') \cdot_Y f(x) \tag{24}$$

holds for all $x \in |X|$ and all atoms $a, a'$ (of the same kind). Clearly the composition of two such functions is another such; and identity functions are morphisms.

Therefore nominal sets and morphisms form a category, which we denote by $\mathcal{N}om$.

**Remark 3.3 (From swapping to permutations).** The following remarks are for readers familiar with the mathematical theory of groups and group actions. It is a standard result of that theory that the group of all permutations of the $n$-element set $\{1, \ldots, n\}$ is isomorphic to the group freely generated by $n-1$ symbols $g_i$ $(i = 1, \ldots, n-1)$, subject to the identities

$$\begin{aligned} (g_i)^2 &= id & \qquad & (i < n) \\ (g_i\, g_{i+1})^3 &= id & & (i < n-1) \\ (g_i\, g_j)^2 &= id & & (j < i-1) \end{aligned}$$

with the generator $g_i$ corresponding to the permutation transposing $i$ and $i+1$. (See for example [16, Beispiel 19.7].) From this fact one can easily deduce that the group of all (kind-respecting) finite permutations of the set of atoms $\mathbb{A}$ is freely generated by the transpositions $(a\, a')$ (with $a, a' \in \mathbb{A}_n$ for some $n \in \mathbb{N}$), subject to the identities

$$\begin{aligned} (a\, a)(a\, a) &= id \\ (a\, a')(a\, a') &= id \\ (a\, a')(b\, b') &= ((a\, a')b \;\; (a\, a')b'\;)(a\, a') \end{aligned}$$

where the atoms $(a\, a')b$ and $(a\, a')b'$ are defined as in equation (22). It follows that if $|X|$ is a set equipped with swapping operations satisfying equations (19)–(21), then these operations extend uniquely to an action of all finite permutations on elements of $|X|$. If $|X|$ also satisfies property (ii) of Definition 3.2, then this action extends uniquely to all (kind-respecting) permutations, finite or not; and the elements of $|X|$ have the finite support property for this action in the sense of [11, Definition 3.3]. These observations form the basis of a proof that *the category $\mathcal{N}om$ of Definition 3.2 is equivalent to the Schanuel topos* [11, Sect. 7], which underlies the universe of FM-sets used in [11].

It is not hard to see that products $X \times Y$ in the category $\mathcal{N}om$ are given simply by taking the cartesian product $\{(x, y) \mid x \in |X| \wedge y \in |Y|\}$ of underlying sets and defining the swapping operations componentwise:

$$(a\, a') \cdot_{X \times Y} (x, y) \triangleq ((a\, a') \cdot_X x, (a\, a') \cdot_Y y).$$

(Clearly $(x, y)$ has the finiteness property in $X \times Y$ required by Definition 3.2(ii), because $x$ has it in $X$ and $y$ has it in $Y$.) Similarly, the terminal object 1 in $\mathcal{N}om$ has a one-element underlying set and (necessarily) trivial swapping operations.

So we can interpret many-sorted first-order signatures in the category $\mathcal{N}om$: sorts $S$ are interpreted as objects $[\![S]\!]$; function symbols $f$, of arity $S_1, \ldots, S_n \longrightarrow S$ say, as morphisms $[\![f]\!] : [\![S_1]\!] \times \cdots \times [\![S_n]\!] \longrightarrow [\![S]\!]$; and relation symbols $R$, of arity $S_1, \ldots, S_n$ say, as subobjects of $[\![S_1]\!] \times \cdots \times [\![S_n]\!]$. Indeed, $\mathcal{N}om$ has sufficient properties to soundly interpret classical first-order logic with equality [1] using the

[1] And much more besides, since it is equivalent to the Schanuel topos, but that will not concern us here.

usual techniques of categorical logic—see [18], or [23, Sect. 5] for a brief overview. In fact, readers unfamiliar with such techniques need not become so just to understand the interpretation of first-order logic in the category of nominal sets, since it is just like the usual Tarskian semantics of first-order logic in the category of sets (at the same time remaining within the world of equivariant properties). For it is not hard to see that the subobjects of an object $X$ in the category $\mathcal{N}om$ are in bijection with the subsets $A \subseteq |X|$ of the underlying set that are equivariant, in the sense that $(a\,a') \cdot_X x \in A$ whenever $x \in A$, for any atoms $a, a'$ (of the same kind). As we mentioned in Sect. 2, the collection of equivariant subsets is closed under all the usual operations of first-order logic and contains equality. So it just remains to explain the interpretation in $\mathcal{N}om$ of the distinctive syntax of Nominal Logic—atoms, swapping and freshness.

**Definition 3.4.** Here is the intended interpretation of atoms, swapping and freshness in the category of nominal sets of Definition 3.2.

**Atoms.** A sort of atoms in a Nominal Logic signature will be interpreted by a *nominal set of atoms* $A_n$ (for some $n \in \mathbb{N}$), which by definition has underlying set $|A_n| = \mathbb{A}_n$ and is equipped with the swapping operations given by

$$(a\,a') \cdot b \triangleq \begin{cases} a & \text{if } b = a' \\ a' & \text{if } b = a \\ b & \text{otherwise} \end{cases}$$

(where $b \in \mathbb{A}_n$ and $a, a' \in \mathbb{A}_m$ for any $m \in \mathbb{N}$). We always assume that distinct sorts of atoms are interpreted by distinct kinds of atoms. (So we are implicitly assuming that signatures contain at most countably many such sorts.)

**Swapping.** Note that by virtue of equation (21), the function $a, a', x \mapsto (a\,a') \cdot_X x$ determines a morphism $A_n \times A_n \times X \longrightarrow X$ in the category $\mathcal{N}om$. This morphism is used to interpret the distinguished function symbol $A, A, S \longrightarrow S$ for swapping, assuming the nominal set of atoms $A_n$ is the interpretation of the sort of atoms $A$ and that $X$ is the interpretation of $S$. Thus

$$[\![(a\,a') \cdot s]\!] = ([\![a]\!]\,[\![a']\!]) \cdot_X [\![s]\!] \quad \text{when } s : S \text{ and } [\![S]\!] = X .$$

**Freshness.** The distinguished relation symbol # of arity $A, S$ for freshness is interpreted as the ‘not in the support of’ relation $(-) \notin \mathit{supp}_X(-)$ between atoms and elements of nominal sets. Thus if the nominal set of atoms $A_n$ is the interpretation of the sort of atoms $A$ and $X$ is the interpretation of the sort $S$, then for terms $a : A, s : S$, the formula $a \mathrel{\#} s$ is satisfied by the interpretation if and only if $[\![a]\!] \notin \mathit{supp}_X([\![s]\!])$, where $\mathit{supp}_X$ is as in equation (23). (It is not hard to see that this is an equivariant subset of $\mathbb{A}_n \times |X|$ and hence determines a subobject of $[\![A]\!] \times [\![S]\!]$ in $\mathcal{N}om$.)

## 4 Nominal Logic Axioms

For simplicity, we will use a Hilbert-style presentation of Nominal Logic: a single rule of Modus Ponens, the usual axiom schemes of first-order logic with equality, plus the axiom schemes for swapping and freshness given in Fig. 1.

*Properties of swapping*

**S1** $(\forall a : A)(\forall x : S)\ (a\,a) \cdot x = x$

**S2** $(\forall a, a' : A)(\forall x : S)\ (a\,a') \cdot (a\,a') \cdot x = x$

**S3** $(\forall a, a' : A)\ (a\,a') \cdot a = a'$

*Equivariance*

**E1** $(\forall a, a' : A)(\forall b, b' : A')(\forall x : S)\ (a\,a') \cdot (b\,b') \cdot x = ((a\,a') \cdot b\ \ (a\,a') \cdot b') \cdot (a\,a') \cdot x$

**E2** $(\forall a, a' : A)(\forall b : A')(\forall x : S)\ b \mathrel{\#} x \Rightarrow (a\,a') \cdot b \mathrel{\#} (a\,a') \cdot x$

**E3** $(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ (a\,a') \cdot f(\vec{x}) = f((a\,a') \cdot \vec{x})$
where $f$ is a function symbol of arity $\vec{S} \longrightarrow S$

**E4** $(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ R(\vec{x}) \Rightarrow R((a\,a') \cdot \vec{x})$
where $R$ is a relation symbol of arity $\vec{S}$

*Properties of freshness*

**F1** $(\forall a, a' : A)(\forall x : S)\ a \mathrel{\#} x \wedge a' \mathrel{\#} x \Rightarrow (a\,a') \cdot x = x$

**F2** $(\forall a, a' : A)\ a \mathrel{\#} a' \Leftrightarrow \neg a = a'$

**F3** $(\forall a : A)(\forall a' : A')\ a \mathrel{\#} a'$
where $A \neq A'$

**F4** $(\forall \vec{x} : \vec{S})(\exists a : A)\ a \mathrel{\#} \vec{x}$

**Notes**

1. $A, A'$ range over sorts of atoms, $S$ ranges over sorts and $\vec{S}$ over finite lists of sorts.
2. In axiom **E3** and **E4**, $(a\,a')\cdot\vec{x}$ indicates the finite list of arguments given by $(a\,a')\cdot x_i$ as $x_i$ ranges over $\vec{x}$.
3. In axiom **F4**, $a \mathrel{\#} \vec{x}$ indicates the finite conjunction of the formulas $a \mathrel{\#} x_i$ as $x_i$ ranges over the list $\vec{x}$.

**Fig. 1.** The axiom schemes of Nominal Logic for freshness and swapping

The following result shows that the axioms in Fig. 1 validate the fundamental assumption mentioned at the start of Sect. 2, namely that all properties expressible in Nominal Logic are invariant under swapping atoms.

**Proposition 4.1 (Equivariance).** *For each term $t$ and formula $\varphi$, with free variables amongst $\vec{x} : \vec{S}$ say, we have*

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ (a\,a') \cdot t(\vec{x}) = t((a\,a') \cdot \vec{x}) \tag{25}$$

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Leftrightarrow \varphi((a\,a') \cdot \vec{x}) \tag{26}$$

*where $t((a\,a') \cdot \vec{x})$ denotes the result of simultaneously substituting $(a\,a') \cdot x_i$ for $x_i$ in $t$ (as $x_i$ ranges over $\vec{x}$) and similarly for $\varphi((a\,a') \cdot \vec{x})$.*

*Proof.* Property (25) follows from axioms **E1** and **E3**, by induction on the structure of the term $t$. For (26) we proceed by induction on the structure of the formula $\varphi$, using standard properties of first-order logic for the induction steps for connectives and quantifiers. Note that by virtue of axiom **S2**, equation (26) holds if and only if

$$(\forall a, a' : A)(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}) \Rightarrow \varphi((a\,a') \cdot \vec{x}) \tag{27}$$

does. So the base case when $\varphi$ is equality follows from the usual axioms for equality, the base case for the freshness predicate # follows from axiom **E2**, and that for relation symbols from axiom **E4** (using (25) in each case). □

**Proposition 4.2 (Soundness).** *The axioms in Fig. 1 are all satisfied by the nominal sets interpretation of atoms, swapping and freshness given in Sect. 3.*

*Proof.* Satisfaction of axioms **S1**–**S3** and **E1** is guaranteed by part (i) of Definition 3.2 (since the swapping action for a nominal set of atoms is given by equation (22)). Satisfaction of axioms **E2** and **F1**–**F3** is a simple consequence of the definition of support in equation (23). Axioms **E3** and **E4** are satisfied because function and relation symbols are interpreted by morphisms and subobjects in the category of nominal sets, which have these equivariance properties. Finally, axiom **F4** is satisfied because the support of an element of a nominal set is a finite subset of the fixed, countably infinite set $\mathbb{A}$ of all atoms. □

Did we forget any axioms? In other words are the axiom schemes in Fig. 1 complete for the intended interpretation in the category of nominal sets? Axiom **F4** says that there is an inexhaustible supply of atoms that are fresh, i.e. not in the support of elements in the current context. This is certainly a consequence of property (ii) of Definition 3.2, which guarantees that elements of nominal sets have finite support. However, that property is ostensibly a statement of weak second order logic, since it quantifies over finite sets of atoms. So we should not expect the first-order theory of Nominal Logic to completely axiomatise the notion of finite support. Example 4.5 confirms this expectation. Before giving it we state a useful property of freshness in Nominal Logic that we need below.

**Proposition 4.3.** *For any term $t$, with free variables amongst $\vec{x} : \vec{S}$ say, we have*

$$(\forall a : A)(\forall \vec{x} : \vec{S})\ a \mathrel{\#} \vec{x} \Rightarrow a \mathrel{\#} t(\vec{x}) \tag{28}$$

*(Recall that $a \mathrel{\#} \vec{x}$ stands for the finite conjunction of the formulas $a \mathrel{\#} x_i$ as $x_i$ ranges over $\vec{x}$.)*

*Proof.* Given any $a : A$ and $\vec{x} : \vec{S}$, by axiom **F4** there is some $a' : A$ with $a' \mathrel{\#} \vec{x}$ and $a' \mathrel{\#} t(\vec{x})$. So if $a \mathrel{\#} \vec{x}$, then by axiom **F1** $(a\,a') \cdot x_i = x_i$ holds for each $x_i$. So since $a' \mathrel{\#} t(\vec{x})$ by choice of $a'$, we have

$$\begin{aligned}
a &= (a\,a') \cdot a' && \text{by axioms } \mathbf{S2} \text{ and } \mathbf{S3}\\
&\mathrel{\#}\ (a\,a') \cdot t(\vec{x}) && \text{by axiom } \mathbf{E2}\\
&= t((a\,a') \cdot \vec{x}) && \text{by (25)}\\
&= t(\vec{x}) && \text{by axiom } \mathbf{F1}
\end{aligned}$$

as required. □

**Corollary 4.4.** *If a Nominal Logic theory contains a closed term $t : A$ with $A$ a sort of atoms, then it is an inconsistent theory.*

*Proof.* Suppose that $A$ is a sort of atoms and that $t : A$ is a term with no free variables. By the above proposition we have $(\forall a : A)\ a \mathrel{\#} t$. Thus $t \mathrel{\#} t$ and by axiom **F2** this means $\neg\, t = t$, contradiction. □

**Example 4.5 (Incompleteness).** Consider the following Nominal Logic theory.

**Sort of atoms:** $A$
**Sorts of data:** $D, N$
**Function symbols:** $o : N$
$s : N \longrightarrow N$
$f : D, N \longrightarrow A$
**Axioms:**

$$\begin{aligned}
&(\forall x : N)\ \neg\, o = s(x)\\
&(\forall x, x' : N)\ s(x) = s(x') \Rightarrow x = x'
\end{aligned}$$

*Claim: any model of this theory in the category of nominal sets satisfies the formula*

$$(\forall y : D)(\exists x, x' : N)\ \neg\, x = x' \wedge f(y, x) = f(y, x') \tag{29}$$

*but that formula cannot be proved in Nominal Logic from the axioms of the theory.*

*Proof of Claim.* Note that in any model of this theory in the category $\mathcal{N}om$, the interpretation of the closed terms $n_k : N$ $(k \in \mathbb{N})$ defined by

$$\begin{cases} n_0 & \triangleq o\\ n_{k+1} & \triangleq s(n_k) \end{cases}$$

are distinct elements $[\![n_k]\!] \in |[\![N]\!]|$ of the nominal set $[\![N]\!]$. Therefore, to see that (29) is satisfied by the model it suffices to show for each $d \in |[\![D]\!]|$ that $[\![f]\!]([\![n_{k_1}]\!], d) = [\![f]\!]([\![n_{k_2}]\!], d) \in |[\![A]\!]|$ holds for some $k_1 \neq k_2 \in \mathbb{N}$. Note that $[\![A]\!]$ is a nominal set of atoms, $\mathbb{A}_n$ say. Suppose to the contrary that all the $[\![f]\!]([\![n_k]\!], d)$

are distinct atoms in $\mathbb{A}_n$. Then since the support $supp_{[\![D]\!]}(d)$ of $d \in |[\![D]\!]|$ is a finite subset of $\mathbb{A}$, we can find $k_1 \neq k_2 \in \mathbb{N}$ so that

$$a_1 \triangleq [\![f]\!]([\![n_{k_1}]\!], d) \quad \text{and} \quad a_2 \triangleq [\![f]\!]([\![n_{k_2}]\!], d)$$

satisfy $a_1, a_2 \notin supp_{[\![D]\!]}(d)$. We also have $a_1, a_2 \notin supp_{[\![N]\!]}(n_k)$ for all $k$ (using (28) and the fact that the terms $n_k$ have no free variables). Hence $a_1, a_2 \notin supp_{\mathbb{A}_n}([\![f]\!]([\![n_k]\!]), d)$ and thus $(a_1\, a_2) \cdot_{\mathbb{A}_n} [\![f]\!]([\![n_k]\!], d) = [\![f]\!]([\![n_k]\!], d)$, for all $k \in \mathbb{N}$. Taking $k = k_1$ and recalling the definition of $a_1$ and $a_2$, we conclude that

$$[\![f]\!]([\![n_{k_2}]\!], d) = a_2 = (a_1\, a_2) \cdot_{\mathbb{A}_n} a_1 = (a_1\, a_2) \cdot_{\mathbb{A}_n} [\![f]\!]([\![n_{k_1}]\!], d) = [\![f]\!]([\![n_{k_1}]\!], d)$$

with $k_1 \neq k_2$, contradicting our assumption that all the $[\![f]\!]([\![n_k]\!], d)$ are distinct.

To see that (29) is not provable in Nominal Logic it suffices to find a model in the usual sense of first-order logic for the axioms of this theory and the axioms in Fig. 1 which does not satisfy (29). We can get such a model by modifying Definition 3.2 by using an *uncountable* set of atoms and sets equipped with swapping actions all of whose elements have *countable* support. More concretely, we get a model $M$ by taking $[\![A]\!]_M$ to be an uncountable set, the set $\mathbb{R}$ of real numbers say; taking $[\![N]\!]_M$ to be a countable subset of this set, the set $\mathbb{N}$ of natural numbers say; and taking $[\![D]\!]_M$ to be the set $\mathbb{R}^{\mathbb{N}}$ of all functions from $\mathbb{N}$ to $\mathbb{R}$ (all such functions are countably supported). The interpretation of the function symbols $o$, $s$ and $f$ are respectively zero, successor ($n \mapsto n+1$) and the evaluation function $\mathbb{R}^{\mathbb{N}} \times \mathbb{N} \longrightarrow \mathbb{R}$ ($d, n \mapsto d(n)$). The interpretation of the swapping operation for sort $A$ is as in equation (22) (i.e. $(r\, r') \cdot_{\mathbb{R}} r'' = (r\, r')r''$ for all $r, r', r'' \in \mathbb{R}$); for sort $N$, swapping is trivial (i.e. $(r\, r') \cdot_{\mathbb{N}} n = n$ for all $r, r' \in \mathbb{R}$ and $n \in \mathbb{N}$); and for sort $D$, it is given by $(r\, r') \cdot_{\mathbb{R}^{\mathbb{N}}} d = \lambda n \in \mathbb{N}.(r\, r') \cdot_{\mathbb{R}} d(n)$. The interpretation of the freshness predicate for sort $A$ is $\neq$; for sort $N$, it is trivial (i.e. $r \# n$ holds for all $r \in \mathbb{R}$ and $n \in \mathbb{N}$); and for sort $D$, $r \# d$ holds if and only if $r \neq d(n)$ for all $n \in \mathbb{N}$. With these definitions one can check that all the axioms are satisfied. However (29) is not satisfied, because the inclusion of $\mathbb{N}$ into $\mathbb{R}$ gives an element $d \in \mathbb{R}^{\mathbb{N}} = [\![D]\!]_M$ for which $n \mapsto [\![f]\!]_M(d, n)$ is injective. □

Even though there is this incompleteness, it appears that the axioms of Nominal Logic are sufficient for a useful theory of names and name-binding along the lines of [11, 9]. The following sections give some evidence for this claim.

## 5 The Freshness Quantifier

We begin by proving within Nominal Logic the characteristic 'some/any' property of fresh atoms (cf. [11, Proposition 4.10]).

**Proposition 5.1.** *Suppose $\varphi$ is a formula with free variables amongst $a : A, \vec{x} : \vec{S}$ (with $A$ a sort of atoms). Then*

$$(\exists a : A)\; a \# \vec{x} \wedge \varphi(a, \vec{x}) \;\Leftrightarrow\; (\forall a : A)\; a \# \vec{x} \Rightarrow \varphi(a, \vec{x}) \qquad (30)$$

*is provable in Nominal Logic.*

*Proof.* If $\varphi(a, \vec{x})$ holds, then by Proposition 4.1 and axiom **S3** we also have $\varphi(a', (a\, a') \cdot \vec{x})$; so if $a \mathrel{\#} \vec{x}$ and $a' \mathrel{\#} \vec{x}$, then axiom **F1** gives $\varphi(a', \vec{x})$. Thus we have the left-to-right implication in (30).

Conversely suppose $(\forall a : A)\ a \mathrel{\#} \vec{x} \Rightarrow \varphi(a, \vec{x})$ holds. For any $\vec{x} : \vec{S}$, using axiom **F4** we can find $a : A$ such that $a \mathrel{\#} \vec{x}$ and hence also satisfying $\varphi(a, \vec{x})$. □

This property of freshness crops up frequently in proofs about syntax with named bound variables (see [19] for example): we choose *some* fresh name with a certain property and later on, in a wider context, we have to revise the choice to accommodate finitely many more constraints and so need to know that we could have chosen *any* fresh name with that property. For this reason it is convenient to introduce a notation that tells us we have this 'some/any' property without mentioning the context of free variables $\vec{x}$ explicitly. (Note that (30) holds for any list $\vec{x}$ so long as it contains the free variables of $\varphi$ other than the atom $a$ being quantified over.

**Definition 5.2 (И-quantifier).** For each formula $\varphi$ and each variable $a : A$ (with $A$ a sort of atoms), define

$$(\text{И}a : A)\ \varphi \triangleq (\exists a : A)\ a \mathrel{\#} \vec{x} \wedge \varphi(a, \vec{x}) \tag{31}$$

where $\vec{x}$ is the list of free variables of $\varphi$ not equal to the variable $a$. (There is no requirement that $a$ actually occur free in $\varphi$.)

We could have formulated Nominal Logic with the И-quantifier as a primitive and the freshness predicate defined from it, since it is not hard to prove from the above definition and the axioms of Nominal Logic that

$$a \mathrel{\#} x \Leftrightarrow (\text{И}a' : A)\ (a\, a') \cdot x = x \tag{32}$$

holds. When taken as primitive, the axioms for И can be derived from the proof rules mentioned in [10, Remark 3.6]. Here we have chosen the presentation with # as primitive to emphasise that Nominal Logic is just a theory within usual first-order logic.

Further evidence of the naturalness of the И-quantifier is the fact that, in the semantics given in Sect. 3, it coincides with a cofiniteness quantifier: $(\text{И}a : A)\ \varphi$ *holds in the nominal sets interpretation if and only if* $\varphi(a)$ *holds for all but finitely many atoms* $a$. See [9] for the development of the properties and applications of the И-quantifier within the setting of FM-set theory.

**Example 5.3 ($\lambda$-Terms mod $\alpha$-equivalence, version 2).** We can simplify some of the axioms of the theory in Example 3.1 using the И-quantifier. Specifically, axiom (13) explaining equality of $\lambda$-abstractions is equivalent to

$$\begin{gathered}(\forall a, a' : \mathit{Var})\, (\forall t, t' : \mathit{Term})\ \mathit{lam}(a, t) = \mathit{lam}(a', t') \Leftrightarrow \\ (\text{И}a'' : \mathit{Var})\ (a''\, a) \cdot t = (a''\, a') \cdot t'\end{gathered} \tag{33}$$

(cf. [11, Lemma 5.1]); axiom (14), which is the structural induction principle for $\alpha$-equivalence classes of $\lambda$-terms, can be reformulated as follows (with the extra free variables in $\varphi$ now implicit):

$$\begin{aligned}&(\forall a : \mathit{Var})\ \varphi(\mathit{var}(a))\\ &\wedge (\forall t, t' : \mathit{Term})\ \varphi(t) \wedge \varphi(t') \Rightarrow \varphi(\mathit{app}(t, t'))\\ &\wedge (\text{И}a : \mathit{Var})(\forall t : \mathit{Term})\ \varphi(t) \Rightarrow \varphi(\mathit{lam}(a, t))\\ &\Rightarrow (\forall t : \mathit{Term})\ \varphi(t)\end{aligned} \tag{34}$$

Finally axiom (18), which is the clause for $\lambda$-abstractions in the structurally recursive definition of capture-avoiding substitution, can be replaced by

$$\begin{aligned}&(\forall t : \mathit{Term})(\forall a : \mathit{Var})\,(\text{И}a' : \mathit{Var})(\forall t' : \mathit{Term})\\ &\qquad \mathit{subst}(t, a, \mathit{lam}(a', t')) = \mathit{lam}(a', \mathit{subst}(t, a, t')).\end{aligned} \tag{35}$$

The following result is the Nominal Logic version of [11, Lemma 6.3], which is used in that paper to introduce notation for 'locally fresh atoms' in FM-set theory. (We discuss extending the term language of Nominal Logic in Sect. 7.)

**Proposition 5.4.** *Suppose that $t$ is a term of sort $S$ with free variables amongst $a : A, \vec{x} : \vec{S}$ (with $A$ a sort of atoms). Then the following is a theorem of Nominal Logic:*

$$\begin{aligned}&(\forall \vec{x} : \vec{S})\ ((\text{И}a : A)\ a \;\#\; t(a, \vec{x})) \Rightarrow\\ &\qquad (\exists! x : S)(\forall a' : A)\ a' \;\#\; \vec{x} \Rightarrow x = t(a', \vec{x})\end{aligned} \tag{36}$$

*(where $\exists!$ means 'there exists a unique ... ' and has the usual encoding in first-order logic).*

*Proof.* Suppose $(\text{И}a : A)\ a \;\#\; t(a, \vec{x})$. So there is some $a : A$ with $a \;\#\; \vec{x}$ and $a \;\#\; t(a, \vec{x})$. Put $x = t(a, \vec{x})$. Clearly, if $x$ has the property $(\forall a' : A)\ a' \;\#\; \vec{x} \Rightarrow x = t(a', \vec{x})$, it is the unique such (since by axiom **F4** there is some $a'$ with $a' \;\#\; \vec{x}$). To see that it does have this property, suppose $a' : A$ satisfies $a' \;\#\; \vec{x}$. Since we want to show that $x = t(a', \vec{x})$, if $a' = a$ then we are done. So suppose $\neg\, a' = a$; thus $a' \;\#\; a$ (by axiom **F2**) and hence $a' \;\#\; t(a, \vec{x})$ by Proposition 4.3. Since $a \;\#\; t(a, \vec{x})$, we have

$$\begin{aligned}x \triangleq t(a, \vec{x}) &= (a\, a') \cdot t(a, \vec{x}) && \text{by axiom } \mathbf{F1}\\ &= t((a\, a') \cdot a, (a\, a') \cdot \vec{x}) && \text{by Proposition 4.1}\\ &= t(a', \vec{x}) && \text{by axioms } \mathbf{S3} \text{ and } \mathbf{F1}\end{aligned}$$

as required. □

## 6 Binding

In Example 5.3, the fact that *lam* is a variable-binding operation is reflected in axiom (33), which explains equality of terms of the form $\mathit{lam}(a, t)$ via a swapping formulation of $\alpha$-conversion (cf. [11, Sect. 2]). Instead of axiomatising binders on a case-by-case basis, we can make a definitional extension of Nominal Logic with a new sort-forming operation for *atom-abstraction* whose intended interpretation is the following construction on nominal sets.

**Definition 6.1 (Nominal set of atom-abstractions).** Given a nominal set $X$ and a nominal set of atoms $A_n$ (cf. Definition 3.4), the *nominal set of atom-abstractions* $[A_n]X$ is defined as follows.

**Underlying set** $|[A_n]X|$ is the set of equivalence classes for the equivalence relation on $\mathbb{A}_n \times |X|$ that relates $(a, x)$ and $(a', x')$ if and only if $(a''\, a) \cdot_X x = (a''\, a') \cdot_X x'$ for some (or indeed any) $a'' \in \mathbb{A}_n$ such that $a'' \notin \mathit{supp}_X(x) \cup \mathit{supp}_X(x') \cup \{a, a'\}$. We write $a.x$ for the equivalence class of the pair $(a, x)$.

**Swapping action** is inherited from that for the product $A_n \times X$:

$$(b\, b') \cdot_{[A_n]X} (a.x) \triangleq a'.x' \quad \text{where } a' = (b\, b')a \text{ and } x' = (b\, b') \cdot_X x.$$

With these definitions one can check that the requirements of Definition 3.2 are satisfied (in particular the support of $a.x$ turns out to be the finite set $\mathit{supp}_X(x) - \{a\}$; cf. Proposition 6.2).

See [11, 9] for the use of this notion of atom-abstraction to treat syntax modulo $\alpha$-equivalence as inductively defined sets (with useful associated structural induction/recursion principles) within the Fraenkel-Mostowski permutation model of set theory. Here we observe that the notion is definable within Nominal Logic. The situation is analogous to the fact that cartesian products are definable within ordinary first-order logic: given sorts $S_1$, $S_2$ and $S$, there is a first-order theory in all of whose models the interpretation of $S$ is isomorphic to the cartesian product of the interpretations of $S_1$ and $S_2$. Indeed there are several such theories; for example, take a function symbol $\mathit{pair} : S_1, S_2 \longrightarrow S$ and axioms

$$\begin{aligned}&(\forall x_1, x_1' : S_1)\,(\forall x_2, x_2' : S_2)\ \mathit{pair}(x_1, x_2) = \mathit{pair}(x_1', x_2') \Rightarrow \\ &\qquad (x_1 = x_1') \wedge (x_2 = x_2') \end{aligned} \tag{37}$$

$$(\forall x : S)(\exists x_1 : S_1)(\exists x_2 : S_2)\ x = \mathit{pair}(x_1, x_2). \tag{38}$$

Within Nominal Logic there is a similar definability result for atom-abstraction sets. Given sorts $A$, $S$ and $S'$ (with $A$ a sort of atoms), and a function symbol $\mathit{abs} : A, S \longrightarrow S'$, the axioms

$$\begin{aligned}&(\forall a, a' : A)\,(\forall x, x' : S)\ \mathit{abs}(a, x) = \mathit{abs}(a', x') \Leftrightarrow \\ &\qquad (\text{И} a'' : A)\ (a''\, a) \cdot x = (a''\, a') \cdot x' \end{aligned} \tag{39}$$

$$(\forall x' : S')(\exists a : A)(\exists x : S)\ x' = \mathit{abs}(a, x) \tag{40}$$

ensure that in the semantics of Sect. 3, the interpretation of $S'$ is isomorphic to $[A_n]X$, where $A_n$ and $X$ are the nominal sets interpreting $A$ and $S$ respectively.

Figure 2 gives an extension of Nominal Logic with atom-abstractions. Axiom **E5** ensures that we still have the crucial equivariance properties of Proposition 4.1 for the extended syntax (and hence also the freshness property of Proposition 4.3). For axiom **A1** we have chosen an equivalent formulation of (39) avoiding the use of the freshness quantifier; as we noted above, this together with axiom **A2** determine the meaning of $[A]S$ and $a.s$ in the category

Add to the syntax of Nominal Logic as follows.

- For each sort of atoms $A$ and each sort $S$, there is a sort of data $[A]S$, called the *sort of $A$-atom-abstractions of $S$*.
- For each sort of atoms $A$ and each sort $S$ there is a distinguished function symbol of arity $A, S \longrightarrow [A]S$ whose effect on terms $a : A$, and $s : S$ we write as the term $a.s$ and pronounce '*abstract $a$ in $s$*'.

Add to the axioms of Nominal Logic the following.

**E5** $(\forall b, b' : A')(\forall a : A)(\forall x : S)\ (b\,b') \cdot (a.x) = ((b\,b') \cdot a).((b\,b') \cdot x)$
**A1** $(\forall a, a' : A)(\forall x, x' : S)\ a.x = a'.x' \Leftrightarrow (a = a' \wedge x = x') \vee (a' \mathrel{\#} x \ \wedge\ x' = (a\,a') \cdot x)$
**A2** $(\forall y : [A]S)(\exists a : A)(\exists x : S)\ y = a.x$

**Fig. 2.** Nominal Logic with atom-abstractions

$\mathcal{N}om$ up to isomorphism. For this reason, the following characterisation of freshness for atom-abstractions is a theorem of the extended Nominal Logic, rather than one of its axioms.

**Proposition 6.2.** *If $A$ and $A'$ are distinct sorts of atoms and $S$ is any sort, then the following formulas are provable in Nominal Logic extended as in Fig. 2.*

$$(\forall a, a' : A)(\forall x : S)\ a' \mathrel{\#} a.x \ \Leftrightarrow\ a' = a \ \vee\ a' \mathrel{\#} x \tag{41}$$
$$(\forall a : A)(\forall a' : A')(\forall x : S)\ a' \mathrel{\#} a.x \ \Leftrightarrow\ a' \mathrel{\#} x \tag{42}$$

*Proof.* In view of axioms **F2** and **F3**, it suffices to prove

$$(\forall a : A)(\forall x : S)\ a \mathrel{\#} a.x \tag{43}$$
$$(\forall a : A)(\forall a' : A')(\forall x : S)\ a' \mathrel{\#} x \ \Rightarrow\ a' \mathrel{\#} a.x \tag{44}$$
$$(\forall a : A)(\forall a' : A')(\forall x : S)\ a' \mathrel{\#} a \ \wedge\ a' \mathrel{\#} a.x \ \Rightarrow\ a' \mathrel{\#} x \tag{45}$$

for all sorts of atoms $A$ and $A'$ (possibly equal).

For (43), given $a : A$ and $x : S$, by axiom **F4** we can find $a' : A$ with $a' \mathrel{\#} a.x$ and hence

$$\begin{aligned} a &= (a\,a') \cdot a' && \text{by axioms } \mathbf{S2} \text{ and } \mathbf{S3} \\ &\mathrel{\#}\ (a\,a') \cdot (a.x) && \text{by axiom } \mathbf{E2} \text{ on } a' \mathrel{\#} a.x \\ &= a'.((a\,a') \cdot x) && \text{by axioms } \mathbf{E5} \text{ and } \mathbf{S3} \\ &= a.x && \text{by axiom } \mathbf{A1}. \end{aligned}$$

For (44), given $a : A$, $a' : A'$ and $x : S$ with $a' \mathrel{\#} x$, we argue by cases according to whether $A$ and $A'$ are the same and whether $a' = a$ or not. If the sorts are the same and $a' = a$, then we have $a' \mathrel{\#} a.x$ by (43); in the other three cases we always have $a' \mathrel{\#} a$ (using axioms **F2** and **F3**); so since $a' \mathrel{\#} a$

and $a' \mathrel{\#} x$, we have $a' \mathrel{\#} a.x$ by Proposition 4.3 (which holds for the extended syntax by virtue of axiom **E5**).

For (45), given $a : A$, $a' : A'$ and $x : S$ with $a' \mathrel{\#} a$ and $a' \mathrel{\#} a.x$, by axiom **F4** we can find $a'' : A'$ with $a'' \mathrel{\#} a$, $a'' \mathrel{\#} x$ and $a'' \mathrel{\#} a.x$. Then

$$\begin{aligned} a.x &= (a'\,a'') \cdot a.x && \text{by axiom } \mathbf{F1} \\ &= ((a'\,a'') \cdot a).(a'\,a'') \cdot x) && \text{by axiom } \mathbf{E5} \\ &= a.((a'\,a'') \cdot x) && \text{by axiom } \mathbf{F1} \end{aligned}$$

and hence $x = (a'\,a'') \cdot x$ by axiom **A1**. Since $a'' \mathrel{\#} x$, we get $a' = (a'\,a'') \cdot a'' \mathrel{\#} (a'\,a'') \cdot x = x$, as required. □

**Example 6.3 ($\lambda$-Terms mod $\alpha$-equivalence, version 3).** We can reformulate Example 5.3 to use atom-abstractions by changing the arity of *lam* to be $[\mathit{Var}]\mathit{Term} \longrightarrow \mathit{Term}$. At the same time, axiom (33) is replaced by a simple injectivity requirement like axioms (11) and (12):

$$(\forall y, y' : [\mathit{Var}]\mathit{Term})\ \mathit{lam}(y) = \mathit{lam}(y') \Rightarrow y = y'. \tag{46}$$

Similarly the disjointness axioms (8) and (9) are replaced by

$$(\forall a : \mathit{Var})(\forall y : [\mathit{Var}]\mathit{Term})\ \neg\, \mathit{var}(a) = \mathit{lam}(y) \tag{47}$$

$$(\forall y : [\mathit{Var}]\mathit{Term})(\forall t, t' : \mathit{Term})\ \neg\, \mathit{lam}(y) = \mathit{app}(t, t') \tag{48}$$

and the exhaustion axiom (10) by

$$\begin{aligned} (\forall t : \mathit{Term})\ & (\exists a : \mathit{Var})\ t = \mathit{var}(a) \\ & \vee (\exists t', t'' : \mathit{Term})\ t = \mathit{app}(t', t'') \\ & \vee (\exists y : [\mathit{Var}]\mathit{Term})\ t = \mathit{lam}(y). \end{aligned} \tag{49}$$

The other axioms alter in straightforward ways to take account of the new arity of *lam*.

The following result is needed in the next section. It shows that atom-abstraction sorts $[A]X$ have a dual nature: their elements $a.x$ embody not only the notion of abstraction as a '(bound variable, body)-pair modulo renaming the bound variable', but also the notion of abstraction as a function (albeit a partial one) from atoms to individuals.

**Proposition 6.4.** *The following formula is provable in Nominal Logic extended as in Fig. 2.*

$$(\forall y : [A]S)(\forall a : A)\ a \mathrel{\#} y \Rightarrow (\exists! x : S)\ y = a.x \tag{50}$$

*(where $\exists!$ means 'there exists a unique ... ' and has the usual encoding in first-order logic).*

*Proof.* The uniqueness part of (50) follows from

$$(\forall a : A)(\forall x, x' : S)\ a.x = a.x' \Rightarrow x = x'$$

which is a corollary of axioms **A1** and **S1**. For the existence part of (50), note that by Proposition 5.1

$$(\forall y : [A]S)(\forall a : A)\ a \mathrel{\#} y \Rightarrow (\exists x : S)\ y = a.x$$

holds if and only if

$$(\forall y : [A]S)(\exists a : A)\ a \mathrel{\#} y \wedge (\exists x : S)\ y = a.x$$

and the latter follows from axiom **A2** and Proposition 6.2 (specifically, property (43)). □

## 7 Choice

In informal arguments about syntax one often says things like '*choose* a fresh name such that ...'. Axiom **F4** in Fig. 1 ensures that we can comply with such directives for Nominal Logic's formalisation of freshness. But it is important to note that *in nominal Logic such choices cannot be made uniformly in the parameters*: it is in general inconsistent with the other axioms to Skolemize **F4** by adding function symbols $\mathit{fresh} : \vec{S} \longrightarrow A$ satisfying $(\forall \vec{x} : \vec{S})\ \mathit{fresh}(\vec{x}) \mathrel{\#} \vec{x}$. Here is the simplest possible example of this phenomenon.

**Proposition 7.1.** *Suppose $A$ is a sort of atoms. The formula*

$$(\forall a : A)(\exists a' : A)\ \neg\, a = a' \tag{51}$$

*is a theorem of Nominal Logic. However, it is inconsistent to assume there is a function that, for each atom, picks out an atom different from it; in other words the Nominal Logic theory with a function symbol $f : A \longrightarrow A$ and the axiom*

$$(\forall a : A)\ \neg\, a = f(a) \tag{52}$$

*is inconsistent.*

*Proof.* The formula (51) is an immediate consequence of axioms **F2** and **F4**. For the second part we show that $(\exists a : A)\ a = f(a)$ is a theorem. First note that by axiom **F4** (with the empty list of parameters $\vec{x}$), there is an atom $a$ of sort $A$.[2] We show that $a = f(a)$. For any $a' : A$, by Proposition 4.3 we have $a' \mathrel{\#} a \Rightarrow a' \mathrel{\#} f(a)$, i.e. (by axiom **F2**) $\neg\, a' = a \Rightarrow \neg\, a' = f(a)$, i.e. $a' = f(a) \Rightarrow a' = a$. Taking $a'$ to be $f(a)$, we get $f(a) = a$. □

This phenomenon is a reflection of the fact that the category $\mathcal{N}\mathit{om}$ of nominal sets fails to satisfy the Axiom of Choice (see [8] for a categorical treatment of choice), which in turn reflects the fact that, by design, the Axiom of Choice

[2] The reader can deduce at this point that the author, being of a category-theoretic bent, is not assuming a formulation of first-order logic that entails that all sorts are non-empty. Possibly empty sorts, like the empty set, have their uses!

fails to hold in the Fraenkel-Mostowski permutation model of set theory [17]. However, there is no problem with principles of *unique* choice. For example, if a Nominal Logic theory has a model in $\mathcal{N}om$ satisfying the sentence

$$(\forall \vec{x} : \vec{S})(\exists! x' : S')\ \varphi(\vec{x}, x') \tag{53}$$

then the theory extended by a function symbol $f : \vec{S} \longrightarrow S'$ and axiom

$$(\forall \vec{x} : \vec{S})\ \varphi(\vec{x}, f(\vec{x})) \tag{54}$$

can also be modelled in $\mathcal{N}om$ (simply because in a cartesian category any subobject satisfying the properties of a single-valued and total relation is the graph of some morphism). Unfortunately a far more common situation than (53) is to have 'conditional unique existence':

$$(\forall \vec{x} : \vec{S})\ \delta(\vec{x}) \Rightarrow (\exists! x' : S')\ \varphi(\vec{x}, x') \tag{55}$$

so that $\varphi(\vec{x}, x')$ is the graph of a *partial* function with domain of definition given by $\delta(\vec{x})$—we have already seen two examples of this, in Propositions 5.4 and 6.4. If the formula (55) is a theorem of a Nominal Logic theory, adding a function symbol $f : \vec{S} \longrightarrow S'$ and axiom

$$(\forall \vec{x} : \vec{S})\ \delta(\vec{x}) \Rightarrow \varphi(\vec{x}, f(\vec{x})) \tag{56}$$

can result in an inconsistent theory. This is because $f$ represents a *total* function from $\vec{S}$ to $S'$. Given terms $\vec{s} : \vec{S}$, even if $\delta(\vec{s})$ does not hold and so (56) cannot be used to deduce properties of the term $f(\vec{s}) : S'$, nevertheless one may be able to use results such as Proposition 4.3 to deduce properties of $f(\vec{s}) : S'$ that lead to inconsistency, especially if $S'$ happens to be a sort of atoms. The simplest possible example of this phenomenon is when $\vec{S}$ is the empty list of sorts and $\delta$ is *false*. In this case formula (55) is trivially a theorem; the Skolemizing function $f$ is a constant of sort $S'$, so if that is a sort of atoms we get inconsistency by Corollary 4.4.

This difficulty with introducing notations for possibly partially defined expressions is masked in [10] by the untyped nature of FM-set theory.[3] That work introduces term-formers for *locally fresh atoms* and for *concretion* of atom-abstractions at atoms, Skolemizing the conditional unique existence formulas of Propositions 5.4 and 6.4. These new forms of term only have a definite meaning when certain preconditions are met. Nevertheless they can be given a semantics as total elements of the universe of FM-sets simply by taking their meaning when the preconditions are not met to be some default element with empty support (the empty set, say). Such a 'hack' is available to us in classical logic when there are enough terms of empty support. One such term is enough in an untyped

[3] It is also masked in the programming language FreshML sketched in [24], which has a richer term language than does Nominal Logic; this is because FreshML features unrestricted fixed point recursion in order to be Turing powerful, and hence naturally contains partially defined expressions.

setting such as FM-set theory. In a many-sorted Nominal Logic theory there is nothing to guarantee that a sort $S$ possesses a term $s : S$ of empty support (i.e. satisfying $(\forall a : A)\ a \ \# \ s$ for all sorts of atoms $A$); indeed Corollary 4.4 shows that sorts of atoms do not possess such terms in a consistent theory. Therefore, to provide Nominal Logic with a richer term language, incorporating such things as terms with locally fresh atoms, concretions of atom-abstractions at atoms and maybe more besides, it seems that one should merge Nominal Logic's novel treatment of atoms and freshness with some conventional treatment of the logic of partial expressions (such as [1, Sect. VI.1] or [26]).

## 8 Related Work

One can classify work on fully formal treatments of names and binding according to the mathematical construct used to model the notion of an abstraction over names:

**Abstractions as (name, term)-pairs.** Here one tries to work directly with parse trees quotiented by alpha conversion; [19] and [27] are examples of this approach. Its drawback is not so much that many tedious details left implicit by informal practice become explicit, but rather that many of these details have to be revisited on a case-by-case basis for each object language. The use of parse trees containing de Bruijn indices [5] is more elegant; but this has its own complications and also side-steps the issue of formalising informal practice to do with named bound variables.

**Abstractions as functions from terms to terms.** The desire to take care of the tedious details of $\alpha$-conversion and substitution once and for all at the meta-level leads naturally to encodings of object-level syntax in a typed $\lambda$-calculus. This is the approach of *higher-order abstract syntax* [22] and it is well-supported by existing systems for machine-assisted reasoning based on typed $\lambda$-calculus. It does not lend itself to principles of structural recursion and induction for the encoded object-language that are particularly straightforward, but such principles have been developed: see [6, 25].

**Abstractions as functions from names to terms.** The *Theory of Contexts* [15] reconciles the elegance of higher-order abstract syntax with the desire to deal with names at the object-level and have relatively simple forms of structural recursion/induction. It does so by axiomatizing a suitable type of names within classical higher order logic. The Theory of Contexts involves a 'non-occurrence' predicate and axioms quite similar to those for freshness in FM-set theory [11] and Nominal Logic. However, 'non-occurrence' in [15] is dependent upon the object language, whereas our notion of freshness is a purely logical property, independent of any particular object syntax. (The same remark applies to the axiomatization of $\alpha$-conversion of $\lambda$-terms in higher order logic in [12]; and to the extension of first-order logic with binders studied in [7].) Furthermore, the use of total functions on names to model abstraction means that the Theory of Contexts is incompatible with the Axiom of Unique Choice (cf. Sect. 7), forcing the theory to have a relational rather than functional feel: see [20]. On the other hand, the Theory of Contexts is able to take advantage of existing machine-assisted infrastructure (namely Coq [4]) quite easily, whereas Gabbay had to work hard to adapt the Isabelle [21] set theory package to produce his Isabelle/FM-sets package [9, Chapter III].

The notion of abstraction that is definable within Nominal Logic (see Sect. 6) captures something of the first and third approaches mentioned above: atom-abstractions are defined to be pairs in which the name-component has been made anonymous via swapping; but we saw in Proposition 6.4 that atom-abstractions also behave like functions, albeit partial ones. Whatever the pros and cons of the various views of name abstraction, at least one can say that, being first-order, Nominal Logic gives a more elementary explanation of names and binding than the work mentioned above; and a more general one, I would claim, because of the independence of the notions of atoms, swapping, freshness and atom-abstraction from any particular object-level syntax.

Nominal Logic gives a first-order axiomatisation of some of the key concepts of FM-set theory—atoms, swapping and freshness—which were used in [11] to model syntax modulo $\alpha$-conversion with inductively defined sets whose structural induction/recursion properties remain close to informal practice. We have seen that, being first-order, Nominal Logic does not give a complete axiomatisation of the notion of *finite support* that underlies the notion of freshness in FM-sets. Nevertheless, the first-order properties of the freshness predicate $(-) \mathrel{\#} (-)$ seem sufficient to develop a useful theory. Indeed, many of the axioms in Fig. 1 arose naturally in Gabbay's implementation of FM-set theory in the Isabelle system [9, Chapter III] as the practically useful properties of finite support. Nominal Logic is just a vehicle for exhibiting those properties clearly. If one wants a single, expressive meta-logic in which to develop the mathematics of syntax, one can use FM-set theory (and its automated support within Isabelle); it is certainly also worth considering developing a version of classical higher order logic incorporating Nominal Logic.

Finally, even if one does not care about the details of Nominal Logic, I think that two simple, but important ideas underlying it are worth taking on board for the practice of operational semantics (be it with pencil-and-paper, or with machine assistance):

– *Name-swapping* $(a\,a') \cdot (-)$ *has much nicer logical properties than renaming* $[a/a'](-)$.
– *The only assertions about syntax we should deal with are ones whose validity is invariant under swapping bindable names.*

Even if one only takes the naïve view of abstractions as (name, term)-pairs, it seems useful to define $\alpha$-conversion and capture-avoiding substitution in terms of name-swapping and to take account of equivariance in inductive arguments. We gave a small illustration of this in Example 2.1. A further example is provided by the work of Caires and Cardelli on modal logic for the spatial structure of concurrent systems [2]; this and the related work [3] make use of the freshness quantifier of Sect. 5. See also [14] for the use of permutative renaming to treat naming aspects of process calculi.

*Acknowledgements.* The work described here draws upon joint work with Gabbay described in [10, 24, 11]. I also gratefully acknowledge conversations about FM-sets with members of the Cambridge *Logic and Semantics Group*, particularly Mark Shinwell and Michael Norrish; and thanks to Keith Wansbrough for designing the freshness relation (#) and quantifier (И) symbols in METAFONT.

## References

[1] M. J. Beeson. *Foundations of Constructive Mathematics.* Ergebnisse der Mathematik und ihrer Grenzgebeite. Springer-Verlag, Berlin, 1985.

[2] L. Caires and L. Cardelli. A spatial logic for concurrency. Draft of 11 April, 2001.

[3] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In S. Abramsky, editor, *Typed Lambda Calculus and Applications, 5th International Conference*, volume 2044 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 2001.

[4] The Coq proof assistant. Institut National de Recherche en Informatique et en Automatique, France. `http://coq.inria.fr/`

[5] N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indag. Math.*, 34:381–392, 1972.

[6] J. Despeyroux, F. Pfenning, and C. Schürmann. Primitive recursion for higher-order abstract syntax. In *Typed Lambda Calculus and Applications, 3rd International Conference*, volume 1210 of *Lecture Notes in Computer Science*, pages 147–163. Springer-Verlag, Berlin, 1997.

[7] G. Dowek, T. Hardin, and C. Kirchner. A completeness theorem for an extension of first-order logic with binders. In S. Ambler, R. Crole, and A. Momigliano, editors, *Mechanized Reasoning about Languages with Variable Binding (MERLIN 20001)*, Proceedings of a Workshop held in conjunction with the *International Joint Conference on Automated Reasoning, IJCAR 2001, Siena, June 2001*, Department of Computer Science Technical Report 2001/26, pages 49–63. University of Leicester, 2001.

[8] P. J. Freyd. The axiom of choice. *Journal of Pure and Applied Algebra*, 19:103–125, 1980.

[9] M. J. Gabbay. *A Theory of Inductive Definitions with $\alpha$-Equivalence: Semantics, Implementation, Programming Language.* PhD thesis, Cambridge University, 2000.

[10] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. In *14th Annual Symposium on Logic in Computer Science*, pages 214–224. IEEE Computer Society Press, Washington, 1999.

[11] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 2001. Special issue in honour of Rod Burstall. To appear.

[12] A. D. Gordon and T. Melham. Five axioms of alpha-conversion. In *Theorem Proving in Higher Order Logics: 9th Interational Conference, TPHOLs'96*, volume 1125 of *Lecture Notes in Computer Science*, pages 173–191. Springer-Verlag, Berlin, 1996.

[13] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques.* Foundations of Computing. MIT Press, 1992.

[14] K. Honda. Elementary structures in process theory (1): Sets with renaming. *Mathematical Structures in Computer Science*, 10:617–663, 2000.

[15] F. Honsell, M. Miculan, and I. Scagnetto. An axiomatic approach to metareasoning on systems in higher-order abstract syntax. In *Twenty-Eighth International Colloquium on Automata, Languages and Programming, ICALP 2001, Crete, Greece, July 2001, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, 2001.

[16] B. Huppert. *Endliche Gruppen I*, volume 134 of *Grundlehren Math. Wiss.* Springer, Berlin, 1967.

[17] T. J. Jech. About the axiom of choice. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 345–370. North-Holland, 1977.

[18] M. Makkai and G. E. Reyes. *First Order Categorical Logic*, volume 611 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin, 1977.

[19] J. McKinna and R. Pollack. Some type theory and lambda calculus formalised. To appear in *Journal of Automated Reasoning*, Special Issue on Formalised Mathematical Theories (F. Pfenning, Ed.), 1998.

[20] M. Miculan. Developing (meta)theory of lambda-calculus in the theory of contexts. In S. Ambler, R. Crole, and A. Momigliano, editors, *Mechanized Reasoning about Languages with Variable Binding (MERLIN 20001)*, Proceedings of a Workshop held in conjunction with the *International Joint Conference on Automated Reasoning, IJCAR 2001, Siena, June 2001*, Department of Computer Science Technical Report 2001/26, pages 65–81. University of Leicester, 2001.

[21] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1994.

[22] F. Pfenning and C. Elliott. Higher-order abstract syntax. In *Proc. ACM-SIGPLAN Conference on Programming Language Design and Implementation*, pages 199–208. ACM Press, 1988.

[23] A. M. Pitts. Categorical logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 5. Algebraic and Logical Structures*, chapter 2. Oxford University Press, 2000.

[24] A. M. Pitts and M. J. Gabbay. A metalanguage for programming with bound names modulo renaming. In R. Backhouse and J. N. Oliveira, editors, *Mathematics of Program Construction. 5th International Conference, MPC2000, Ponte de Lima, Portugal, July 2000. Proceedings*, volume 1837 of *Lecture Notes in Computer Science*, pages 230–255. Springer-Verlag, Heidelberg, 2000.

[25] C. Schürmann. *Automating the Meta-Theory of Deductive Systems.* PhD thesis, Carnegie-Mellon University, 2000.

[26] D. S. Scott. Identity and existence in intuitionistic logic. In M. P. Fourman, C. J. Mulvey, and D. S. Scott, editors, *Applications of Sheaves, Proceedings, Durham 1977*, volume 753 of *Lecture Notes in Mathematics*, pages 660–696. Springer-Verlag, Berlin, 1979.

[27] R. Vestergaard and J. Brotherson. A formalised first-order confluence proof for the $\lambda$-calculus using one-sorted variable names. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA 2001, Utrecht, The Netherlands, May 2001, Proceedings*, volume 2051 of *Lecture Notes in Computer Science*, pages 306–321. Springer-Verlag, Heidelberg, 2001.