THE BULLETIN OF SYMBOLIC LOGIC
Volume 17, Number 2, June 2011

FOUNDATIONS OF NOMINAL TECHNIQUES: LOGIC AND SEMANTICS OF VARIABLES IN ABSTRACT SYNTAX

MURDOCH J. GABBAY

Abstract. We are used to the idea that computers operate on numbers, yet another kind of data is equally important: the syntax of formal languages, with variables, binding, and alpha-equivalence. The original application of nominal techniques, and the one with greatest prominence in this paper, is to reasoning on formal syntax with variables and binding.

Variables can be modelled in many ways: for instance as numbers (since we usually take countably many of them); as links (since they may 'point' to a binding site in the term, where they are bound); or as functions (since they often, though not always, represent 'an unknown').

None of these models is perfect. In every case for the models above, problems arise when trying to use them as a basis for a fully formal mechanical treatment of formal language. The problems are practical—but their underlying cause may be mathematical.

The issue is not whether formal syntax exists, since clearly it does, so much as what kind of mathematical structure it is. To illustrate this point by a parody, logical derivations can be modelled using a Gödel encoding (i.e., injected into the natural numbers). It would be false to conclude from this that proof-theory is a branch of number theory and can be understood in terms of, say, Peano's axioms. Similarly, as it turns out, it is false to conclude from the fact that variables can be encoded e.g., as numbers, that the theory of syntax-with-binding can be understood in terms of the theory of syntax-without-binding, plus the theory of numbers (or, taking this to a logical extreme, purely in terms of the theory of numbers). It cannot; something else is going on. What that something else is, has not yet been fully understood.

In nominal techniques, variables are an instance of names, and *names are data*. We model names using urelemente with properties that, pleasingly enough, turn out to have been investigated by Fraenkel and Mostowski in the first half of the 20th century for a completely different purpose than modelling formal language. What makes this model really interesting is that it gives names distinctive properties which can be related to useful logic and programming principles for formal syntax.

Since the initial publications, advances in the mathematics and presentation have been introduced piecemeal in the literature. This paper provides in a single accessible document an updated development of the foundations of nominal techniques. This gives the reader easy access to updated results and new proofs which they would otherwise have to search across two or more papers to find, and full proofs that in other publications may have been elided. We also include some new material not appearing elsewhere.

Received March 18, 2009.

Key words and phrases. Nominal techniques, logic and set theory, nominal abstract syntax, atoms-abstraction, names, variable binding, inductive syntax up to binding, alpha-equivalence.

Thanks to an anonymous referee and to Alexander Kurz.

1079-8986/11/1702-0001/$7.90

Contents

§1. Introduction. This paper overviews the logical and semantic foundations of *nominal techniques*. Nominal techniques were developed with Pitts and introduced in the author's thesis [23]. The outlines of the mathematical foundations of nominal techniques were apparent in 1999 and were presented in a conference paper [36] and in 2001 in a journal paper [37]. Papers using nominal techniques now number over a hundred [54].

If there is a single idea behind nominal techniques, it is to let names inhabit a denotation directly as a form of data (set theorists should think of *urelemente*). That is, the 'x' in $\lambda x.r$, $\forall x.\phi$, $\int_x f(x)dx$, and $\nu x.P$ has an independent denotational reality. This x is, in a mathematical sense that we will make formal, a 'real thing': a name.

The initial application of nominal techniques (and still, arguably, the most important) is to provide a denotation for inductive datatypes of abstract syntax up to α-equivalence—datatypes which admit structural induction and recursion principles while at the same time being intuitively a *quotient* of syntax by α-equivalence (see Theorem 5.18 and Corollary 5.19).

Compare this for example with *de Bruijn indexes* [14], *higher-order abstract syntax* [55], or Bourbaki's *boxes and links* notation [5, Section 1], which can also be applied to represent syntax with binding; they do not make any commitment to names having a denotational reality.

Indeed, an explicit commitment to an *absence* of names may be made. For example Bourbaki write x and y but emphasise "L'assemblage désigné par $\tau_x(A)$ *ne contient donc pas* x", which roughly translates as "So the string denoted by $\tau_x(A)$ *does not contain* x". There have been attempts to construct non-trivial syntaxes for logic, foundations, and programming in which variables need not even tangentially appear, e.g., combinator syntax S and K (instead of λ-calculus) [3], or $\mathcal{L}^\times$ the formalisation of set theory without variables [63, Chapter 3].

Yet names and binding are so widespread that they must surely reflect some mathematical structure that awards them an independent mathematical existence. Interpreting names as urelemente certainly does that. As it turns out, the resulting mathematics is surprisingly rich.

In the several years we have worked with this material, our understanding of the core ideas has changed. An account of this journey is strewn across a history of documents including [23, 37, 26, 33, 30, 34, 48, 27, 16]. This document is our attempt to gather the threads and tell the story as we now understand it. We hope this paper will offer an account of the basis of a useful and interesting field of enquiry.

1.1. A worked example. We consider an example of the kind of problem we address in this paper; it is drawn from material in Section 5, which fleshes out this sketch to full detail. Why is 'abstract syntax with binding' an issue?

Inductively define a datatype of trees representing untyped λ-calculus syntax not up to α-equivalence. Thus, given variable symbols $a, b, c, \ldots$ we define terms by:

- If a is a variable symbol then a is a term.
- If t and s are terms then ts is a term ('t applied to s').
- If a is a variable symbol and s is a term then $\lambda a.s$ is a term ('lambda a, s').[1]

Terms are labelled trees. We obtain an inductive principle on terms:

$$\text{If}\quad \begin{array}{ll} \forall a. & \phi(a) \\ \forall s, s'. & (\phi(s) \wedge \phi(s')) \Rightarrow \phi(ss') \\ \forall a.\forall s. & \phi(s) \Rightarrow \phi(\lambda a.s) \end{array} \quad \text{then} \quad \forall s.\phi(s).$$

As is standard, we define *free variables* $fv(s)$ inductively by:

$$fv(a) = \{a\}, \qquad fv(ss') = fv(s) \cup fv(s'), \qquad fv(\lambda a.s) = fv(s) \backslash \{a\}.$$

We can then try to define a 'substitution function' as follows:

$$\begin{aligned} a[a \mapsto s''] &= s'', \\ b[a \mapsto s''] &= b, \\ (ss')[a \mapsto s''] &= (s[a \mapsto s''])(s'[a \mapsto s'']), \\ (\lambda a.s)[a \mapsto s''] &= \lambda a.s, \\ (\lambda b.s)[a \mapsto s''] &= \lambda c.(s[b \mapsto c][a \mapsto s'']) \qquad (b \in fv(s''),\ c \text{ fresh}), \\ (\lambda b.s)[a \mapsto s''] &= \lambda b.(s[a \mapsto s'']) \qquad\qquad\qquad (b \notin fv(s'')). \end{aligned}$$

Here a, b, and c range over distinct variable symbols. There are two problems with this:

- The definition is not inductive because in the clause for $(\lambda b.s)[a \mapsto s'']$, $s[b \mapsto c]$ is not a subterm of $\lambda b.s$.
- It is not a function unless we make some fixed but arbitrary choice of fresh c.

There are various ways to fix this. We can work inductively on a measure of *size*, prove a lemma that $s[b \mapsto c]$ has the same size as s, and then prove that the definition above specifies a function. For each finite set of atoms S we can make some fixed but arbitrary choice of 'fresh' atom c such that $c \notin S$ and appeal to that choice where we wrote 'c fresh' above.

[1] This is expressed compactly by a BNF definition as follows: $s ::= a \mid ss \mid \lambda a.s$.

Some authors define a *simultaneous* substitution, as in [62, 17]; this avoids, but does not solve, some problems, and we pay a certain price in complexity. Others change the datatype entirely, as in the de Bruijn indexes and Higher-Order Abstract Syntax approaches [14, 4, 55].

What becomes apparent is that we are using a datatype whose native inductive principle is not as useful as we might first suppose. This is not a good situation to be in if our goal is to find a mathematics to represent and reason about variable symbols in abstract syntax with binders, and this we must do, if we are to apply formal methods to help design programming and theorem-proving on sentences in formal languages with binders.

The problem is pervasive and does not go away once we have defined substitution. Consider that we may define α-*equivalence* $=_\alpha$ as the least equivalence relation such that:

$$\frac{}{a =_\alpha a} \qquad \frac{s =_\alpha s' \quad t =_\alpha t'}{st =_\alpha s't'}$$

$$\frac{s =_\alpha s'}{\lambda a.s =_\alpha \lambda a.s'} \qquad \frac{s[a \mapsto c] =_\alpha s'[b \mapsto c]}{\lambda a.s =_\alpha \lambda b.s'} \ (c \text{ fresh})$$

Here again the same features surface; choices of fresh atoms, and terms above the line that are smaller than, but are not subterms of, terms below the line.

In this paper we present a mathematics which, amongst other things, allows us to define datatypes of syntax with binding that are truly inductive, and to program on them and reason about them in a way that is very close to informal practice, while remaining mathematically completely rigorous.

1.2. Comments on the connections with previous work. For the benefit of the reader who may be familiar with the author's previous work, we will make brief comments on how what we do here relates with that work:

- *Cumulative hierarchy model first*. We open our technical development in Section 2 with a von Neumann style cumulative sets hierarchy. This takes names (modelled by a collection of *atoms* $\mathbb{A}$) as urelemente and builds up using powersets.

  The reason we start with a cumulative hierarchy model is that this presentation should be accessible to everybody, in the sense that everybody knows, or at least is inclined to believe that they know, what a set is. It also helps us when we derive meta-mathematical properties as properties of first-order logic, like the *equivariance* results outlined in Section 4.

  The sets presentation, the introduction of names as urelemente in a cumulative hierarchy, and the use of familiar mathematical language, is also intended to be in line with classic constructions in the foundations of mathematics, such as those of ordered pairs, Dedekind cuts, graphs of functions, von Neumann ordinals, and so on.

- *We do not insist on finite support*. Unlike in [23, 36, 37], we do not insist on *finite support* (Definition 2.16) when we construct our cumulative hierarchy.

  Nominal techniques are compatible with the existence of non-finitely-supported elements and with the axiom of choice; more on this in the body of the paper.

  We want to dispel a misconception that 'nominal techniques force a change in mathematical foundations' or that 'nominal techniques are inconsistent with the axiom of choice'. We brought this on ourselves when we wrote sentences like 'using a different set theory ... is not to be taken lightly' in [37, Section 1]. It is not necessary to insist on finite support throughout the sets universe.
- *Axiomatic set theory.* As distinct from [23, 36, 37] we de-emphasise logical foundations, that is, axiomatic set theory. For example, in Sections 2 and 3 we develop atoms-abstraction directly, without formalising the meta-language in which we develop it or the foundational axioms we assume.

  However, the burden of proving results without having *equivariance*, a fundamental meta-mathematical observation, eventually becomes too much. We introduce equivariance in Section 4 and make frequent use of it from then on. Even so, we observe equivariance as a property of first-order logic with equality $=$ and set membership $\in$ (a sufficient formal language in which to conduct the mathematics in this paper).

  The full logical foundational view of nominal techniques, represented by the two axiomatic set theories Zermelo–Fraenkel set theory with atoms and Fraenkel–Mostowski set theory, appears in Section 10. This brings us full circle in the sense that the cumulative hierarchy model with which we open our development in Section 2 is also the canonical cumulative hierarchy model of the set theory with atoms with which we conclude it in Section 10.
- *Full proof that datatypes* are *up to α-equivalence.* We give a full proof that 'nominal' inductive syntax *does* yield datatypes of 'syntax-up-to-binding'; see Theorem 5.18 and Corollary 5.19. This was stated in [36] as Theorem 5.1 and in [37] as Theorem 6.2, but only outline proofs were given. As it turns out, finding the right set of lemmas to prove this result nicely and in detail, is not entirely trivial (Definition 5.15 and Lemma 5.16 seem to be useful).
- *Use of equivariance.* As a general rule, working with name-carrying syntax quotiented by α-equivalence looks easy when viewed from afar, and it becomes difficult when viewed close up. To this, nominal techniques offer two related but distinct answers:

– use the atoms-abstraction introduced in [37] to build the datatype purely inductively (material in Section 3) *or*
– use equivariance Theorem 4.4 to permute names as needed, without losing the inductive hypothesis.

Atoms-abstraction is developed in this document, in Section 3.

We use equivariance often in this document (as Theorem 4.4, or as one of its corollaries like Theorems 4.7 or 6.5), but elsewhere we apply equivariance in the specific case of large inductive proofs of properties by induction on formal syntax; we use it to permute names, usually to avoid some form of accidental name-capture, while preserving inductive hypotheses. From the point of view of this paper, these examples are relevant case studies.

The use of permutative renamings of variable symbols predates 'nominal' work; see for example [49, Subsection 9.2]. The work initiated by the author and Pitts systematised the treatment of permutations and put it in a foundational context (for example, the mathematics in this paper).

There is more to equivariance than 'using permutations'. Equivariance lets us preserve inductive hypotheses 'for free'. This is important in implementations, and its application in discursive proof—that is, to get a short, elegant proof of a theorem in a published paper—is also new. The interested reader can view the proof of Lemma 8.3 in [25]. To our knowledge this is the first use of equivariance to rename variable symbols in a discursive inductive proof on abstract syntax. See also for example the proofs of Theorems 5.1, 5.2, 5.3, and Lemma 5.7 in [34].

In short, equivariance offers working mathematicians and computer scientists a rigorous 'short-cut' to handle renaming in their papers—even if they do not use atoms-abstraction, and even if they are not using a theorem-prover or mechanised mathematics. We have certainly found it useful on several occasions, as referenced in the previous paragraph.

- *New proofs and more general results.* Our presentations of atoms-abstraction in Section 3, and of the atoms-abstraction type-former in Section 8, are completely rewritten relative to [23, 37].

  This paper includes some new results (notably: Theorem 3.6; most of the results in Subsection 8.3; many of the results up to and including Theorem 8.18; and Subsection 9.5).

  This paper also includes some results which are familiar but appear here in a slightly more general form (for example, Theorems 8.14 and 8.20). There is also a little bit of precise commentary on the connection with Mostowski's work from the 1930s; in Remark 2.22 we make connections betweeen some results fundamental to this paper, and results in a paper by Mostowski, with exact references.[2]

[2]Thanks to an anonymous referee for suggesting we do this.

Finally, most of the familiar results in this paper have re-worked proofs.

This constitutes a substantive advance over the original presentations, it assembles bits and pieces from other developments into a new integrated discourse, and also we add new elements specific to this paper.

- *Nominal techniques as a general methodology, compatible with numerous frameworks.* Our constructions can be realised in a variety of contexts. It may be worth making a point of sketching some of them in a brief list.

  This list is not exhaustive, but it demonstrates that 'nominal techniques' can exist conveniently in more than one framework. Here are some of the places in which the ideas reported on in this paper have been usefully applied as part of other research:
  - logics (as in nominal logic [56], spatial logic [7], or one-and-a-half level logic [34]);
  - programming (as in FreshOCaml [59], αProlog [11], multi-level λ-calculi [32, 22] and proof-terms [35]);
  - semantics such as domains [61] and categories and set theories (we consider some here in Sections 9 and 10)
  - game theory [1], and of course
  - theorem provers [65, 2].

  Further applications are cited online [54]. We do not detail these applications, but in Sections 9 and 10 we put the various categories and set theories arising from nominal techniques side-by-side with each other and with the cumulative hierarchy we began with in Section 2. This attempts to give some overview of the more general mathematical semantic places in which the basic ideas of nominal techniques can be realised.[3]

In this paper we present the core ideas; representing finitary syntax up to α-conversion. We discuss some extensions to this mathematics in Section 11.

For us, nominal techniques are not only about representing abstract syntax up to variable binding; they are a general mathematics of names. Their

[3]We have in mind a referee who had one of our papers rejected on the grounds that 'nominal techniques only have equivariant functions' and that we had used in our paper a non-equivariant function.

The referee had confused arrows in the category of nominal sets NOM (Definition 9.17) with functions in the underlying set of an exponential and more generally had confused the use of a particular category with an ontological commitment.

This convinced us that something like Sections 9 and 10 is necessary. Any given paper may emphasise one particular presentation (set theory, cumulative hierarchy, sets category, presheafs ...) if only to keep within space limits. This may give the reader of that paper, unfamiliar with the broader context, a mistakenly impression that nominal techniques 'are' the category of nominal sets, or that they 'are' FM set theory, or whatever.

usefulness extends beyond their original intended application to abstract syntax. Further applications are investigated in the author's other papers, and as listed above, the author is just one of several researchers writing papers in which these ideas are used in different and unique ways.

1.3. Notational conventions. We will use the following conventions:

- α and β range over ordinals (Subsection 2.1).
- $\mathbb{A}$ denotes the set of atoms (Definition 2.3). π and τ range over finitely-supported permutations of atoms and $\mathbb{P}$ denotes the set of all finitely-supported permutations of atoms (Definition 2.13).
- ϕ ranges over predicates that can be expressed in the language of ZFA set theory (see Sections 4 and 10).
- χ ranges over functions that can be expressed using predicates ϕ, as outlined in Definition 4.5.
- $\mathcal{U}$ is the cumulative hierarchy of all elements (Definition 2.4). $\mathcal{F}$ ranges over functions from $\mathcal{U}$ to $\mathcal{U}$ (not necessarily only functions expressible as a χ). $\mathcal{HFS}$ is the subclass of $\mathcal{U}$ of hereditarily finitely supported elements (Definition 10.6).

  The intuition is that something written in calligraphic font is 'a very large collection'.
- $x, y, z, \dots$ range over elements in $\mathcal{U}$.
- $a, b, c, \dots$ range over distinct atoms (so 'a and b' means 'any two distinct atoms'; we call this the *permutative convention*).
- $U, V, W, X, Y, Z, \dots$ range over elements in $\mathcal{U}$ that are sets (equivalently, over elements that are not atoms).
- A, B, C range over finite sets of atoms.
- $\hat{x}$ and $\hat{y}$ range over elements that are atoms-abstractions (Remark 3.9).
- f, g, h range over function-sets (Definition 2.10).
- t and t' range over elements of Λ (Definition 5.2). s and s' range over elements of Λ_{nc} (Definition 5.3).
- Sans-serif font indicates a category; ZFASet, FMSet, NOMSet, and NOM.
- $\mathbb{X}, \mathbb{Y}$ range over nominal sets (Definition 9.17).
- F and G are functors between categories.

§2. A cumulative hierarchy with names.

2.1. Some background: ordinals. Ordinals were introduced by Cantor [8]. A concise introduction by Johnstone is available [42]. A Wikipedia article [70] deserves mention for its clarity and accessibility. Ordinals are a large field, so we devote a few lines to outlining what we need for this paper, with sketch definitions.

Definition 2.1.
- A relation $<$ is *transitive* when $x < y$ and $y < z$ imply $x < z$.
- A relation is *antisymmetric* when $x < y$ implies *not* $y < x$.

- A relation on some collection $\mathcal{X}$ (which may or may not be a set) is *total* when $x < y$, $x = y$, or $y < x$ for all x and y in $\mathcal{X}$.
- Write $x > y$ for $y < x$. A relation is *well-founded* when there are no infinite descending $<$-chains (that is, when $x_1 > x_2 > x_3 > \cdots$ is impossible).
- A *well-ordering* is a well-founded transitive antisymmetric total relation.

DEFINITION 2.2. Let *ordinals* be the collection of isomorphism classes of well-ordered collections.

Ordinals are naturally ordered by the relation 'is an initial segment of'; $\alpha \leq \beta$ when there is an order-preserving bijection of a well-ordered collection in α with an initial segment of a well-ordered collection in β.

Examples of ordinals (in ascending order) are:

- The ordinal 1, pictured as '$\bullet$'.
- The ordinal 2, pictured as '$\bullet < \bullet$'.
- The ordinal ω, pictured as '$\bullet < \bullet < \bullet < \cdots$' (the natural numbers, ordered in their natural order, are in this equivalence class).
- The ordinal $\omega + 1$, pictured as '$(\bullet < \bullet < \bullet < \cdots) < \bullet$' (a countable list of elements 'going on forever', plus one more element greater than all others).

2.2. The cumulative hierarchy universe $\mathcal{U}$. $\mathcal{U}$ is the standard *cumulative hierarchy* model of Zermelo–Fraenkel set theory with atoms (Section 10). This construction is due to Zermelo and von Neumann [71, 68]. $\mathcal{U}$ is a sufficiently rich structure that it can encode standard mathematical constructions (see Subsection 2.3), and it can serve as a model for nominal techniques.

DEFINITION 2.3. Fix a countably infinite set $\mathbb{A}$ of *atoms*. $a, b, c, \ldots$ will range over distinct elements of $\mathbb{A}$; we call this a *permutative convention*.

Our permutative convention is designed to model informal practice. When we write 'λx.λy.xy' or '$\forall$x.$\exists$y.x $\neq$ y' we normally take it for granted that 'x' and 'y' denote a pair of distinct variable symbols. It is convenient to reflect this by letting the variables we use to range specifically over atoms, range over distinct atoms.

DEFINITION 2.4. We define a collection of *elements* $\mathcal{U}$ in the style of von Neumann [41] by ordinal induction as follows:

1. $\mathcal{U}_0 = \mathbb{A}$.
2. If $\alpha < \beta$ and $U \in \mathcal{U}_\alpha$ then $U \in \mathcal{U}_\beta$.
3. If $U \subseteq \bigcup_{\alpha<\beta} \mathcal{U}_\alpha$ then $U \in \mathcal{U}_\beta$.

Write $x \in \mathcal{U}$ for '$x \in \mathcal{U}_\alpha$, for some ordinal α', and read this as x *is an element*. We let $\mathcal{U}$ be the collection of all elements, that is, $\mathcal{U}$ is the collection of all x such that $x \in \mathcal{U}$.

We will call an element that is not an atom a *set*.[4]
Examples are illustrated in Figure 1.

$$
\begin{array}{c}
a \in \mathcal{U}_0 \qquad b \in \mathcal{U}_0 \qquad \varnothing \subseteq \mathcal{U}_0 \qquad \mathbb{A} \subseteq \mathcal{U}_0 \\
\{a\} \in \mathcal{U}_1 \qquad \{a, b\} \in \mathcal{U}_1 \qquad \varnothing \in \mathcal{U}_1 \qquad \mathbb{A} \in \mathcal{U}_1 \\
\{\{a\}, \{a, b\}\} \in \mathcal{U}_2 \qquad \{\varnothing\} \in \mathcal{U}_2 \qquad \mathbb{A} \cup \{\mathbb{A}\} \in \mathcal{U}_2 \\
\vdots \\
\mathbb{N} = \{0, 1, 2, \dots\} \in \mathcal{U}_\omega \\
\vdots
\end{array}
$$

FIGURE 1. Example sets in the cumulative hierarchy.

DEFINITION 2.5. Let x be an element. Define the *rank* of x, write it $\mathit{rank}(x)$, to be the least ordinal i for which $x \in \mathcal{U}_i$ is true.

For example:

$$\mathit{rank}(a) = 0, \qquad \mathit{rank}(\varnothing) = 1, \qquad \mathit{rank}(\{\varnothing\}) = 2, \qquad \mathit{rank}(\mathbb{A}) = 1.$$

More on rank in Subsection 4.1 and Lemma 5.5.

REMARK 2.6. Some might find it useful to draw elements of $\mathcal{U}$ as trees (with unordered daughters):

- The tree representing an atom a is a node labelled by a.
- The tree representing the empty set $\varnothing$ is a node with no label.
- The tree representing a set X is a node whose daughters are the trees representing $x \in X$.

(Of course, there are trees that do not represent elements of $\mathcal{U}$; e.g., trees with repeated daughters like a node with two daughters and no labels or other nodes.)

Note that trees modelling sets may be infinitely broad, and the collection of paths down a tree may have no finite bound on its length (an example is $\mathbb{N}$ in Definition 2.8). However, any single path down a tree, from the root node towards the leaves, is finitely long; this gives some intuitive justification for the rank function from Definition 2.5, and for rank induction. $\Diamond$

2.3. Standard constructions in $\mathcal{U}$: numbers, pairs, disjoint sums, and functions. $\mathcal{U}$ is rich enough in structure to model much useful mathematics. For completeness we include a brief sketch of the constructions. We see that they are unaffected by the presence (or absence) of atoms.

[4] If X is a set then $X = \{x \mid x \in X\}$. This is not the case of atoms. For example $a \neq \{x \mid x \in a\} = \varnothing$.

DEFINITION 2.7. Let x and y be elements. Define the *ordered pair* (x, y) by

$$(x, y) = \{\{x\}, \{x, y\}\}.$$

This model of (x, y) is known as the *Kuratowski* implementation.

Let X and Y be sets. Define the *product set* $X \times Y$ by

$$X \times Y = \{(x, y) \mid x \in X,\ y \in Y\}.$$

DEFINITION 2.8. Define a model of numbers as sets by $0 = \{\}$ and $n+1 = n \cup \{n\}$. Write $\mathbb{N} = \{0, 1, 2, 3, \dots\} \in \mathcal{U}$. Also, write $\mathbb{N}_1 = \{1, 2, 3, \dots\} \in \mathcal{U}$.

Define a model of boolean truth-values by $\mathbb{B} = \{0, 1\}$.

DEFINITION 2.9. Let x and y be elements. Define

$$\mathit{inl}(x) = (\varnothing, x) \text{ and } \mathit{inr}(y) = (\{\varnothing\}, y).$$

Let X and Y be sets. Define the *disjoint sum* $X + Y$ by

$$X + Y = \{\mathit{inl}(x) \mid x \in X\} \cup \{\mathit{inr}(y) \mid y \in Y\}.$$

Functions are implemented as *graphs* $f = \{(x, f(x)\}$:

DEFINITION 2.10. Let f be an element. Call f a *function-set* when

$$\forall z \in f.\exists x.\exists y.z = (x, y)$$

(so f is a set of pairs) and

$$\forall (x, y) \in f.\forall (x', y') \in f.x = x' \Rightarrow y = y'.$$

Define $f(x)$ to be the unique y such that $(x, y) \in f$ if this y exists, and $f(x)$ is undefined otherwise.

We let f, g range over elements that are function-sets.

DEFINITION 2.11. Let f be a function-set. Define:

$$\mathit{dom}(f) = \{x \mid \exists y.(x, y) \in f\},$$
$$\mathit{img}(f) = \{y \mid \exists x.(x, y) \in f\}.$$

We call $\mathit{dom}(f)$ the *domain* of f, and we call $\mathit{img}(f)$ the *image* of f.

DEFINITION 2.12. Write $X \to Y$ for the set

$$\{f \mid f \text{ a function set},\ \mathit{dom}(f) = X,\ \mathit{img}(f) \subseteq Y\}.$$

That is, $X \to Y$ is the set of functions mapping elements of X into Y (Y is the *range* of f, considered as an element of $X \to Y$). As is standard, we call this the *function-space* from X to Y.

We may use λ-term notation to express function-sets. For example if $\chi(z_1, \dots, z_n, x)$ expresses a mapping (intuitively, $z_1, \dots, z_n$ are 'parameters' and x is the 'argument') then we write

$$\lambda x \in X.\chi(z_1, \dots, z_n, x) \text{ for the set } \{(x, \chi(z_1, \dots, z_n, x)) \mid x \in X\}.$$

We will always make clear whether we are talking about a 'real' function, or its model as function-sets in $\mathcal{U}$; $\lambda x.exp$ is a 'real' function (which takes x and returns exp, for whatever meaning exp may have), whereas $\lambda x \in X.exp$ denotes a function-set.

2.4. Permutations and support. We now begin to explore the structure of $\mathcal{U}$ from a 'nominal' perspective. We start by considering how to permutatively rename atoms in elements of $\mathcal{U}$ (Definition 2.13). The permutation action reflects the fact that atoms have no internal structure, so 'one atom will do as well as any other atom'—later, we turn this into a theorem when we use the permutation action to define and prove equivariance in Subsection 4.2.

It is also possible to define a notion of dependency of an element on a set of atoms. We call this key notion *support* (Definition 2.16). Perhaps the first non-trivial observation here is that if an element has a finite supporting set of atoms, then it has a unique *least* finite supporting set of atoms. This is Theorem 2.21; this property is one way in which nominal techniques are distinguished from related approaches to names and binding based on presheaves [19] (see also Remark 9.13).

The reader can think of the support of an element as a generalisation of the notion of 'free names in', generalising this notion from abstract syntax trees to all sets. As discussed in the Introduction, not every element in $\mathcal{U}$ has finite support, but we will be most interested in the ones that do.

Definition 2.13. A *permutation* π is a bijection on $\mathbb{A}$ such that $\{a \mid \pi(a) \neq a\}$ is finite (we say that π has *finite support*). π, π', τ will range over permutations. We also use the following notation:

- Write *id* for the *identity permutation*, so $id(a) = a$ always.
- Write $\circ$ for functional composition. So $(\pi\circ\pi')(a) = \pi(\pi'(a))$.
- Write π^{-1} for the inverse of π, so $\pi\circ\pi^{-1} = id = \pi^{-1}\circ\pi$.
- Write $\mathbb{P}$ for the set of all permutations.

Definition 2.14. We define a *permutation action* inductively by:

$$\boxed{\pi\cdot a = \pi(a) \qquad \pi\cdot X = \{\pi\cdot x \mid x \in X\} \quad (X \text{ not an atom})}$$

Lemma 2.15. *$id\cdot x = x$ and $\pi'\cdot(\pi\cdot x) = (\pi'\circ\pi)\cdot x$.*
In words, permutation is a group action on $\mathcal{U}$ (see also (2) of Definition 9.15).

Proof. By a routine induction on $\mathcal{U}$.

- The case of an atom a. From Definition 2.14 it is immediate that $id\cdot a = a$ and $\pi'\cdot(\pi\cdot a) = \pi'(\pi(a)) = (\pi'\circ\pi)\cdot a$.
- The case of a set X. From Definition 2.14 and the inductive hypothesis for every $x \in X$. ⊣

Definition 2.16. Let A be a finite set of atoms.

- Write $\mathit{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}$.

- Say that A *supports* x when $\pi{\cdot}x = x$ for all $\pi \in \mathit{fix}(A)$.
- Say x has *finite support* when some finite A supporting x exists.
- Define $\mathit{supp}\,(x)$ the *support* of x by

$$\boxed{\mathit{supp}\,(x) = \{a \mid \forall A.A \text{ a finite set of atoms supporting } x \Rightarrow a \in A\}}$$

  if x has finite support, and $\mathit{supp}\,(x)$ is undefined otherwise.[5]
- Write $a\#x$ when $a \notin \mathit{supp}\,(x)$ and call *a fresh for x*.[6] Write $a\#x, y, z$ for '$a\#x$ and $a\#y$ and $a\#z$', and so on. ⊣

REMARK 2.17. It is important to realise that $\mathit{supp}\,(x)$ is not equal to $x \cap \mathbb{A}$. $\mathit{supp}\,(x)$ contains the 'conspicuous' atoms of x. An atom is 'inconspicuous' when it can be permuted for other inconspicuous atoms without changing x. Finite support ensures that there are plenty of these inconspicuous atoms around.

It is also important to realise that an atom can be conspicuous by its absence as well as its presence. For instance, $\mathit{supp}\,(\mathbb{A}\backslash\{a\}) = \{a\}$. It is a fact that $a \notin \mathbb{A}\backslash\{a\}$, and that makes it conspicuous relative to the other atoms in $\mathbb{A}$. ◇

REMARK 2.18. Not every element of $\mathcal{U}$ has finite support. Make a fixed but arbitrary choice of bijection of $\mathbb{A}$ with the natural numbers $\{0, 1, 2, 3, 4, 5, \dots\}$. Let $\mathit{comb} \subseteq \mathbb{A}$ be the element corresponding under this bijection with the even numbers $\{0, 2, 4, \dots\}$.

comb contains 'every other atom' $\{a, c, e, g, \dots\}$. It forms a 'comb' through the set of all atoms, where the 'teeth' of the comb are the atoms in *comb*, and the 'spaces between the teeth' are the atoms not in *comb*.

There is no finite $A \subseteq \mathbb{A}$ such that if $\pi \in \mathit{fix}(A)$ then $\pi{\cdot}\mathit{comb} = \mathit{comb}$. ◇

THEOREM 2.19. *A supports x if and only if $\pi{\cdot}A$ supports $\pi{\cdot}x$.*
As an immediate corollary, $\pi{\cdot}\mathit{supp}\,(x) = \mathit{supp}\,(\pi{\cdot}x)$.

PROOF. π is invertible so it suffices to show that if A supports x then $\pi{\cdot}A$ supports $\pi{\cdot}x$. Suppose A supports x and suppose $\tau \in \mathit{fix}(\pi{\cdot}A)$. By easy calculations $\pi^{-1}\circ\tau\circ\pi \in \mathit{fix}(A)$. Therefore $(\pi^{-1}\circ\tau\circ\pi){\cdot}x = x$ and so $\tau{\cdot}(\pi{\cdot}x) = \pi{\cdot}x$. ⊣

[5] Using standard notation, we can write

$$\mathit{supp}\,(x) = \bigcap\{A \mid A \text{ a finite set of atoms supporting } x\}.$$

[6] In the original conference paper [36] $a\#x$ was read 'a is *apart from* x', but this terminology seems to be obsolete now. For example by the time we wrote the journal version [37] we were reading # as *fresh for*.

DEFINITION 2.20. If π is a permutation and $A \subseteq \mathbb{A}$ is a set of atoms, write $\pi|_A$ for the partial function such that

$$\pi|_A(a) = \begin{cases} \pi(a) & \text{if } a \in A, \\ \text{undefined} & \text{if } a \in \mathbb{A}\backslash A. \end{cases}$$

THEOREM 2.21. *Let x be any element. If A and B are finite and support x then so does $A \cap B$. As a corollary*:

1. *If x has a finite supporting set then it has a least finite supporting set and this is equal to $supp\,(x)$.*
2. *If $\pi|_{supp\,(x)} = \pi'|_{supp\,(x)}$ then $\pi{\cdot}x = \pi' x$.*

PROOF. The corollary follows by elementary calculations and the definition of support in Definition 2.16.

Suppose τ fixes $A \cap B$ pointwise. We must show $\tau{\cdot}x = x$. Write

$$K \text{ for } \{a \mid \tau(a) \neq a\}.$$

Choose an injection $\imath$ of $B\backslash A$ into $\mathbb{A} \setminus (A \cup B \cup K)$. Define a permutation π by

$$\begin{aligned} \pi(a) &= \imath(a), && a \in B\backslash A, \\ \pi(\imath(a)) &= a, && a \in B\backslash A, \\ \pi(b) &= b, && b \notin (B\backslash A) \cup img(\imath). \end{aligned}$$

Note that $\pi{\circ}\pi = id$, so $\pi = \pi^{-1}$. π fixes A pointwise so $\pi{\cdot}x = x$. Also $\pi{\circ}\tau{\circ}\pi$ fixes B pointwise so $(\pi{\circ}\tau{\circ}\pi){\cdot}x = x$. We apply π to both sides, use Lemma 2.15, and simplify, and conclude that $\tau{\cdot}x = x$ as required. ⊣

REMARK 2.22. The constructions above have been seen before, though written in a different language (literally) and with very different aims: they correspond with certain constructions in a paper by Mostowski [52] (also in English translation [53]).

It is quite interesting to trace how these ideas appear in [52, 53]. Our notation is very different from Mostowski's, and Mostowski's papers embed the results relevant to us in a broader and more complex argument.[7] Therefore, for the reader's convenience we will briefly trace through a few of the common points, giving precise references.

Definition 2.14 corresponds with Definition 30 in [53]. Lemma 2.15 corresponds with Results 33 and 36. The first part of Theorem 2.19 corresponds with Result 41. The first part of Theorem 2.21 corresponds with Result 88 and the first part of the corollary (the construction of *supp* (x)) corresponds with Result 101. ◇

[7] In [52, 53] it is shown by an argument based on constructing a model with a permutation action, that in ZF plus an axiom that every set can be linearly ordered, we cannot derive that every set can be well-ordered.

2.5. Swappings. A particular kind of permutation will be useful later:

Definition 2.23. Write $(a\ b)$ for the *swapping* permutation which transposes a and b. That is:

$$\boxed{\begin{aligned}(a\ b)(a) &= b,\\ (a\ b)(b) &= a,\\ (a\ b)(c) &= c, \qquad \text{all other } c.\end{aligned}}$$

Remark 2.24. [37, Proposition 3.4] proves 'if A supports x and $\{b \mid (b\ a)\cdot x = x\}$ is infinite, then $A\setminus\{a\}$ supports x'. For sets with finite support this is equivalent to Theorem 2.21, but the statement of Theorem 2.21 is more 'topological' and less 'pointwise'.

Theorem 2.21 makes clearer the correspondence with pullback-preserving functions (cf. Theorem 9.14). The form of Theorem 2.21 is probably also more useful for considering generalisations of the notion of support, e.g., to infinite sets [26]. $\Diamond$

Remark 2.25. Note that in [37] and [36], a permutation is taken to be *any* bijection on atoms—not just bijections with finite support. The design decision made in Definition 2.13, to use finitely-supported bijections, gives us the advantage of easier proofs: for example, in Theorem 2.21 K must be finite so we can conveniently build ι.

In practice there need be no difference in power between the 'finitely-supported permutation' mathematics and the 'all bijections' mathematics. We will be most interested in elements x with a finite supporting set. For any bijection f, there exists a finitely-supported bijection π with the same action on $supp(x)$. The action of f away from x will never be important to us (that is, we will only ever care about a finite part of f).

Restricting to finitely-supported permutations in the first place has the advantage that we can conveniently choose 'fresh atoms', so we have preferred this slightly stronger formulation of the definitions. $\Diamond$

Lemma 2.26. *For each finite $A \subseteq \mathbb{A}$, $fix(A)$ is a group, and is generated as a group by swappings $(a\ b)$ such that $a, b \notin A$.*[8]

As a corollary taking $A = \varnothing$, $\mathbb{P}$ is a group and is generated as a group by swappings.

Proof. It easy to verify that $fix(A)$ is a group.

We now show that $fix(A)$ is generated as a group by swappings. Suppose $\pi \in fix(A)$. Recall from Definition 2.13 that $\{a \mid \pi(a) \neq a\}$ is finite; we work by induction on the size of this set. If $\{a \mid \pi(a) \neq a\}$ is empty then $\pi = id$ and we are done. Otherwise, suppose a is such that $\pi(a) \neq a$. We

[8] As is standard, 'generated as a group' means 'every element can be made by finite combinations with group composition and group inverse, starting from the generators'.

consider $\pi' = \pi \circ (\pi^{-1}(a)\ a)$. It is easy to verify that $\pi' \in \mathit{fix}(A)$ and that the inductive hypothesis is applicable to π'. The result follows. $\dashv$

We find Theorem 2.21 to be consistently the most convenient formulation of the notion of support. On the other hand, the later Theorem 6.2 is elegant, since we can take advantage of the many good properties of the 'new' quantifier И introduced by the author and Pitts in [37], plus finite support (most notably, Theorem 6.5). Yet Theorem 2.21 is impredicative, and Theorem 6.2 uses a quantifier which is less well-known than, say, $\forall$ and $\exists$. We therefore mention Theorem 2.27, which is a simple and direct way to characterise the atoms in $\mathit{supp}\,(x)$:

THEOREM 2.27. *Let x be an element with finite support.*
Then $a \in \mathit{supp}\,(x)$ if and only if $\{b \in \mathbb{A} \mid (b\ a){\cdot}x = x\}$ is finite.

PROOF. Suppose $a \notin \mathit{supp}\,(x)$. By Theorem 2.21 $(b\ a){\cdot}x = x$ for all $b \notin \mathit{supp}\,(x)$.
Conversely suppose $a \in \mathit{supp}\,(x)$. By Theorem 2.19 $(b\ a){\cdot}\mathit{supp}\,(x) = \mathit{supp}\,((b\ a){\cdot}x)$ and it follows that if $b \notin \mathit{supp}\,(x)$ then $(b\ a){\cdot}x \neq x$. $\dashv$

REMARK 2.28. Suppose x has a finite supporting set. For the reader's convenience we collect three ways of detecting atoms in $\mathit{supp}\,(x)$:

- $a \in \mathit{supp}\,(x)$ when $a \in A$ for every finite $A \subseteq \mathbb{A}$ such that A supports x (Theorem 2.21).
- $a \in \mathit{supp}\,(x)$ when $(b\ a){\cdot}x = x$ for only finitely many b (Theorem 2.27).
- $a \in \mathit{supp}\,(x)$ when И$b.(b\ a){\cdot}x \neq x$ (Theorem 6.2). The И-quantifier (pronounced 'new quantifier') is defined in Definition 6.1. $\diamond$

2.6. Permutation and support acting on the standard constructions. We can now consider the behaviour of permutation and support on standard mathematical constructions, such as ordered pairs and functions. Theorem 2.29 seems to capture a lot of this behaviour; for example we use it to give a one-line proof of the important Corollary 2.30. We also consider the permutation action on function-sets (Theorem 2.33).

THEOREM 2.29. *Suppose X is a set, all of whose elements have finite support. If $\bigcup\{\mathit{supp}\,(x) \mid x \in X\}$ is finite then $\mathit{supp}\,(X)$ exists and*

$$\mathit{supp}\,(X) = \bigcup\{\mathit{supp}\,(x) \mid x \in X\}.$$

In particular if X is finite (so that $\bigcup\{\mathit{supp}\,(x) \mid x \in X\}$ is a finite union of finite sets and so is finite) then $\mathit{supp}\,(X) = \bigcup\{\mathit{supp}\,(x) \mid x \in X\}$.

PROOF. Suppose that $\bigcup\{\mathit{supp}\,(x) \mid x \in X\}$ is finite. We prove two set inclusions:

- $\mathit{supp}\,(X) \subseteq \bigcup\{\mathit{supp}\,(x) \mid x \in X\}$. If $\bigcup\{\mathit{supp}\,(x) \mid x \in X\}$ is finite then the result follows by Theorem 2.21 and the fact that $\pi{\cdot}X = \{\pi{\cdot}x \mid x \in X\}$.

- $\bigcup\{supp\,(x) \mid x \in X\} \subseteq supp\,(X)$. Suppose $x \in X$ and $a \in supp\,(x)$. Choose fresh b (so $b\#X$ and $b\#x'$ for every $x' \in X$). By Theorem 2.19 $supp\,((b\ a)\cdot x) = (b\ a)\cdot supp\,(x)$. Since X has no element y such that $b \in supp\,(y)$, we know that $(b\ a)\cdot x \neq X$ and by Theorem 2.21 it must be that $a \in supp\,(X)$. ⊣

Corollary 2.30. *Let x and y be elements with finite support. Then*

$$\begin{aligned} supp\,((x, y)) &= supp\,(x) \cup supp\,(y), \\ supp\,(inl(x)) &= supp\,(x), \\ supp\,(inr(y)) &= supp\,(y). \end{aligned}$$

Proof. Directly from Theorem 2.29. ⊣

Remark 2.31. Corollary 2.30 is a specific corollary of a general result (Theorem 2.29). It can be obtained in another way: by Theorem 4.7 (proved later, but there is no circularity) $supp\,((x, y)) \subseteq supp\,(x) \cup supp\,(y)$, and also, observing the existence of first- and second-projection functions, by Theorem 4.7 also $supp\,(x) \subseteq supp\,((x, y))$ and $supp\,(y) \subseteq supp\,((x, y))$.

However, Theorem 2.29 does not follow from Theorem 4.7. These are two distinct results. ◇

Remark 2.32. If X is not finite then Theorem 2.29 fails. For example $supp\,(\mathbb{A}) = \varnothing$ but

$$\bigcup\{supp\,(a) \mid a \in \mathbb{A}\} = \mathbb{A} \neq \varnothing.$$

Similarly $supp\,(\mathbb{A}\backslash\{c\}) = \{c\}$ but

$$\bigcup\{supp\,(a) \mid a \in \mathbb{A} \land a \neq c\} = \mathbb{A}\backslash\{c\} \neq \{c\}. \qquad \diamond$$

Theorem 2.33. *Suppose $f \in X \to Y$ is a function-set. Then $\pi\cdot f$ is a function-set in $\pi\cdot X \to \pi\cdot Y$, and it represents the function*

$$\lambda x \in \pi\cdot X.\pi\cdot(f(\pi^{-1}\cdot x)).$$

This is the conjugation action.

Proof. π acting on f as a set is

$$\{(\pi\cdot x, \pi\cdot(f(x))) \mid x \in X\}.$$

The result is quickly derived by noting from Lemma 2.15 that $\pi^{-1}\cdot\pi\cdot x = x$ always.[9] ⊣

[9]This also follows from equivariance (Theorem 4.4), developed later: function-set application is equivariant so $\pi\cdot(f(x)) = (\pi\cdot f)(\pi\cdot x)$.

§3. Atoms-abstraction. We mentioned in Section 2 that support generalises 'free names of' from abstract syntax trees to all elements. It turns out that α-binding admits a similar generalisation, to all elements with finite support. The construction works by building, in the sets universe $\mathcal{U}$, a set which is visibly an 'α-equivalence class'. In Subsection 3.1 we consider the theory of α-equivalence classes in somewhat more generality than was considered in previous work [37, 23]. Definition 3.8 describes the specific instance which will be of most interest us, which is abstraction of an element with finite support by a single atom (written $[a]x$), and Subsection 3.2 develops its theory.

It may be worth briefly noting how generalisations of support and abstraction are useful. Generalising 'free names of' to a notion of finite support applicable to elements, allows us to deal with this notion more abstractly and also lets us choose a fresh atom for a complex non-syntactic structure (e.g., an element representing a function, or a game). Generalising α-equivalence similarly lets us model name-binding abstractly, and for complex non-syntactic structures. This leads to the new inductive principles described in [37] (also see Section 7) and so to its implementations, and also for example to denotational semantics ([60] and [1] are examples).

3.1. Generalised α-equivalence classes. $\mathcal{U}$ admits a model of α-abstraction given by forming equivalence classes of 'renamed variants' of an element with finite support. For example,

$$\{\{a,c\},\ \{b,c\},\ \{d,c\},\ \{e,c\},\ \{f,c\},\ \dots\}$$

models 'α-abstract a in $\{a,c\}$' (and 'α-abstract b in $\{b,c\}$', and 'α-abstract d in $\{d,c\}$'). Also for example,

$$\begin{array}{l}\big\{\ \{a,b,c\},\ \{d,b,c\},\ \{e,b,c)\},\ \{f,b,c\},\ \dots \\ \quad \{a,d,c\},\ \{b,d,c\},\ \{e,d,c\},\ \{f,d,c\},\ \dots \\ \quad \{a,e,c\},\ \{b,e,c\},\ \{d,e,c\},\ \{f,e,c\},\ \dots\big\}\end{array}$$

represents 'abstract a and b *simultaneously* in $\{a,b,c\}$'.

The two examples above are equal to $\{a,c\}^{\circlearrowleft}_{\{c\}}$ and $\{a,b,c\}^{\circlearrowleft}_{\{c\}}$ respectively, in the notation developed in Definition 3.1.

In this subsection we make the model formal using *permutation orbits* in Definition 3.1 and the subsequent results. The immediate application of permutation orbits is to abstract atoms one at a time; see the atoms-abstraction operator described in Subsection 3.2 and its application to model syntax with binding in Definition 5.2 in Section 5. However the generalisation is of interest as 'the mathematics behind the atoms-abstraction operator'—it also gives an elegant proof-method, it may be useful in future work (for example to give a direct model of simultaneous quantification), and indeed we have used the notion of permutation orbits in recent past work, to consider substitution as an action on $\mathcal{U}$ (rather than on abstract syntax) [30].

Our main result here is Theorem 3.6, which states a form of soundness that permutation orbits express a notion of simultaneous α-abstraction.

DEFINITION 3.1. Suppose $A \subseteq \mathbb{A}$ is finite. Write

$$\boxed{u\circlearrowleft_A \text{ for } \{\pi\cdot u \mid \pi \in \mathit{fix}(A)\}.}$$

We call this the *permutation orbit* of u under permutations fixing A pointwise.

EXAMPLE 3.2. For example:

- $\{a, b\}\circlearrowleft_{\{a,c\}} = \{\{a, y\} \mid y \in \mathbb{A}\backslash\{a, c\}\}$.
- $a\circlearrowleft_{\varnothing} = \mathbb{A}$.
- $b\circlearrowleft_{\{a\}} = \mathbb{A}\backslash\{a\}$.
- $a\circlearrowleft_{\{a\}} = \{a\}$.
- $u\circlearrowleft_{\varnothing} = \{\pi\cdot u \mid \text{all } \pi\}$.

LEMMA 3.3. *If $\pi \in \mathit{fix}(A)$ then $u\circlearrowleft_A = (\pi\cdot u)\circlearrowleft_A$.*

PROOF. $u\circlearrowleft_A$ is the permutation orbit of u under the action of $\mathit{fix}(A)$. $\mathit{fix}(A)$ is a group by Lemma 2.26. The result follows. $\dashv$

LEMMA 3.4. $\pi\cdot(u\circlearrowleft_A) = (\pi\cdot u)\circlearrowleft_{\pi\cdot A}$. (*The action is pointwise.*)

(Lemma 3.4 may also be obtained by equivariance, Theorem 4.4.)

PROOF. To prove $\pi\cdot(u\circlearrowleft_A) \subseteq (\pi\cdot u)\circlearrowleft_{\pi\cdot A}$ take $x \in \pi\cdot(u\circlearrowleft_A)$. Then $x = \pi\cdot(\tau\cdot u)$ for some $\tau \in \mathit{fix}(A)$. It follows using Lemma 2.15 that $x = (\pi\circ\tau\circ\pi^{-1})\cdot\pi\cdot u$. It is a fact that $\pi\circ\tau\circ\pi^{-1} \in \mathit{fix}(\pi\cdot A)$, so we are done.

The proof that $(\pi\cdot u)\circlearrowleft_{\pi\cdot A} \subseteq \pi\cdot(u\circlearrowleft_A)$ is similar. $\dashv$

Lemma 3.5 is useful for proving Theorem 3.6:

LEMMA 3.5. *Suppose $A \subseteq \mathbb{A}$ is finite and u is an element with finite support. Suppose that $\mathit{supp}\,(u)\backslash A \neq \varnothing$. Then*

$$a \in A \textit{ if and only if } (\forall u' \in u\circlearrowleft_A.a \in \mathit{supp}\,(u')) \vee (\forall u' \in u\circlearrowleft_A.a \notin \mathit{supp}\,(u')).$$

PROOF. We reason by cases:

- Suppose $a \in \mathit{supp}\,(u) \cap A$. By Theorem 2.19 $a \in \mathit{supp}\,(u')$ for all $u' \in u\circlearrowleft_A$.
- Suppose $a \in A\backslash\mathit{supp}\,(u)$. By Theorem 2.19 $a \notin \mathit{supp}\,(u')$ for all $u' \in u\circlearrowleft_A$.
- Suppose $a \notin A$. Since $\mathit{supp}\,(u) \setminus A \neq \varnothing$ by Theorem 2.19 there exist some $u' \in u\circlearrowleft_A$ such that $a \in \mathit{supp}\,(u')$, and some $u' \in u\circlearrowleft_A$ such that $a \notin \mathit{supp}\,(u')$.

The result follows. $\dashv$

Theorem 3.6 is the technical heart of the proof of Theorem 3.11, but it stands on its own as a result describing the interaction of support with taking permutation orbits $u\circlearrowleft_A$.

THEOREM 3.6. *Suppose $A \subseteq \mathbb{A}$ is finite and u is an element with finite support. Then*:

- *If $supp\,(u) \subseteq A$ then $supp\,(u\circlearrowleft_A) = supp\,(u)$.*
- *$supp\,(u\circlearrowleft_A) \subseteq A$ always.*

As a corollary,

$$supp\,(u)\backslash A \neq \varnothing \text{ implies } supp\,(u\circlearrowleft_A) = A.$$

PROOF.
- If $supp\,(u) \subseteq A$ then by the definition of support, if $\pi \in fix(A)$ then $\pi{\cdot}u = u$ and $u\circlearrowleft_A = \{u\}$. It is easy to verify that $supp\,(\{u\}) = supp\,(u)$.
- Suppose that $\pi \in fix(A)$. We use Lemmas 3.3 and 3.4:

$$\pi{\cdot}(u\circlearrowleft_A) = (\pi{\cdot}u)\circlearrowleft_{\pi\cdot A} = (\pi{\cdot}u)\circlearrowleft_A = u\circlearrowleft_A.$$

By Theorem 2.21 the result follows.

Now suppose $supp\,(u)\backslash A \neq \varnothing$. By part 2 of this result $supp\,(u\circlearrowleft_A) \subseteq A$. It remains to prove the reverse inclusion.

Choose any $a \in A$ and any $b \notin A$. By Lemma 3.5 we know that

$$(\forall u'.u' \in u\circlearrowleft_A \Rightarrow a \in supp\,(u')) \vee (\forall u'.u' \in u\circlearrowleft_A \Rightarrow a \notin supp\,(u')).$$

By Definition 2.14 and Theorem 2.19 we also know that

$$(\exists u'.u' \in (b\ a){\cdot}(u\circlearrowleft_A) \wedge a \in supp\,(u')) \\ \wedge (\exists u'.u' \in (b\ a){\cdot}(u\circlearrowleft_A) \wedge a \notin supp\,(u')).$$

It follows that $u\circlearrowleft_A \neq (b\ a){\cdot}(u\circlearrowleft_A)$, so by Theorem 2.21 it follows that $a \in supp\,(u\circlearrowleft_A)$. ⊣

EXAMPLE 3.7. It may be useful to illustrate Theorem 3.6 with two examples:

- *An example where $supp\,(u) \subseteq A$.*

$$a\circlearrowleft_{\{a,b\}} = \{a\}.$$

Here $supp\,(a) \subseteq \{a, b\}$ and $supp\,(a\circlearrowleft_{\{a,b\}}) = supp\,(a)$.
- *An example where $supp\,(u) \not\subseteq A$.*

$$a\circlearrowleft_{\{b\}} = \{a,\ c,\ d,\ e,\ f,\ \dots\} = \mathbb{A}\backslash\{b\}.$$

Here $supp\,(a)\backslash\{b\} = \{a\}$ and $supp\,(\mathbb{A}\backslash\{b\}) = \{b\}$.

3.2. Atoms-abstraction. We now come to *atoms-abstraction* $[a]x$, a central component of the model of syntax-with-binding [37, 23]. Permutation orbits generalise α-abstraction. We demonstrate this in Definition 3.8 by using them to build it:

DEFINITION 3.8. Let x be an element with finite support.
Define $[a]x$ the *(atoms-)abstraction* of a in x, by

$$\boxed{[a]x = (a, x)^{\circlearrowleft}_{supp\,(x)\setminus\{a\}}.}$$

REMARK 3.9. By convention, variables $\hat{x}$ and $\hat{y}$ will range over elements that are atoms-abstractions. That is, if we write $\hat{x}$ then it means we are assuming that

$$\exists a.\exists x.\ x \text{ has finite support and } \hat{x} = [a]x. \qquad \diamond$$

REMARK 3.10. In this subsection and the next we characterise atoms-abstraction in three ways. For the reader's convenience we assemble the characterisations with very brief comments. Full definitions and notation follow.

- $[a]x = (a, x)^{\circlearrowleft}_{supp\,(x)\setminus\{a\}}$ (Definition 3.8).
  This places $[a]x$ as an instance of a more general construction.
- $[a]x = \{(a, x)\} \cup \{(b, (b\ a)\cdot x) \mid b\#x\}$ (Lemma 3.13).
  This enumerates the elements in $[a]x$ as a set and makes clear in what sense it can be considered functional.
- $\hat{x} = \{(a, \hat{x}@a) \mid a\#\hat{x}\}$ (Corollary 3.18).
  ($@$ is 'concretion', the destructor for $[a]$-. See Lemma 3.16.) This enumerates the elements of an atoms-abstraction in terms of all possible actions of the corresponding destructor. $\diamond$

THEOREM 3.11. *Let x be an element with finite support. Let a be any atom. Then*

$$supp\,([a]x) = supp\,(x)\setminus\{a\}.$$

PROOF. $supp\,((a, x)) = supp\,(x) \cup \{a\}$. By definition

$$[a]x = (a, x)^{\circlearrowleft}_{supp\,(x)\setminus\{a\}}.$$

The result follows by Theorem 3.6. $\dashv$

LEMMA 3.12. *Let x and y be elements with finite support. Then*:

1. $[a]x = [b]y$ *if and only if* $b\#x$ *and* $(b\ a)\cdot x = y$.
2. $[a]x = [b]y$ *if and only if for some fresh* c *(so* $c\#x, y$*) it is the case that* $(c\ a)\cdot x = (c\ b)\cdot y$.

PROOF. Suppose that $[a]x = [b]y$. By Theorem 3.11 we know $b\#x$. By Theorem 2.21 if $\pi \in fix(supp\,(x)\setminus\{a\})$ then $\pi\cdot x = (\pi(a)\ a)\cdot x$. Since $(b, y) \in [a]x$ it follows that $(b\ a)\cdot x = y$. Conversely suppose that $b\#x$ and $(b\ a)\cdot x = y$. By Theorem 3.11 and Theorem 2.21 $(b\ a)[a]x = [a]x$. The result follows by Lemma 3.4.

For the second part, suppose $[a]x = [b]y$ and suppose that c is fresh (so $c\#x, y$). By the first part $b\#x$ and $(b\ a)\cdot x = y$, so $(c\ b)\cdot(b\ a)\cdot x = (c\ b)\cdot y$. The result follows by Lemma 2.15 and Theorem 2.21. Now suppose that $(c\ a)\cdot x = (c\ b)\cdot y$. By Theorem 2.19 we can calculate that $b\#x$. Also $(b\ c)\cdot(c\ a)\cdot x = (b\ c)\cdot(c\ b)\cdot y = y$ and by Theorem 2.21 it follows that $(c\ b)\cdot y = y$. ⊣

Our definition of atoms-abstraction matches that of [37, Lemma 5.1]:

LEMMA 3.13. $[a]x = \{(a, x)\} \cup \{(b, (b\ a)\cdot x) \mid b\#x\}$.

PROOF. From Definition 3.8 and Theorem 2.21. ⊣

REMARK 3.14. The reader may wonder why we define $[a]x$ to be $(a, x)\circlearrowleft_{supp(x)\setminus\{a\}}$—would not $x\circlearrowleft_{supp(x)\setminus\{a\}}$ do as well?

We could do this, but we prefer not to for several reasons. One reason is historical: we maintain compatibility with the definition in [37]. Another reason is that $(a, x)\circlearrowleft_{supp(x)\setminus\{a\}}$ is more suggestive because it makes $[a]x$ look like (a, x), which it is designed to emulate in inductive datatypes (see Λ and Λ_{nc} in Section 5).

Finally, there is a practical reason: every representative

$$z \in (a, x)\circlearrowleft_{supp(x)\setminus\{a\}}$$

carries information about the 'abstracted atom' which can be extracted using a first projection, also, our definition *always* returns an infinite equivalence class whereas $x\circlearrowleft_{supp(x)\setminus\{a\}}$ may be equal to $\{x\}$ if $a\#x$, which is inconvenient in the sense that, if we are just given $\{x\}$ without further comment, it is not clear whether we mean '$[a]x$' where it happens that $a\#x$, or 'singleton x'.[10] ◇

LEMMA 3.15. *Let $\hat{x}$ be an atoms-abstraction (with finite support).*[11] *Then*

$$(a, x) \in \hat{x} \text{ if and only if } \hat{x} = [a]x.$$

PROOF. $(a, x) \in [a]x$ by Definition 3.8; this is the right-to-left implication. For the left-to-right implication we reason by cases:

- Suppose $(a, x) \in [a]x'$. By Definition 3.8 $x = x'$ and we are done.
- Suppose $(a, x) \in [b]y$. By Definition 3.8 $b\#x$ and $y = (b\ a)\cdot x$. The result follows by part 1 of Lemma 3.12. ⊣

[10] There is some analogy here with *anamorphisms* [47], which can be thought of as functors annotated with extra information about choices of representative; analogously, we can write functions on an atoms-abstraction $[a]x$ by writing functions on representatives (a, x), which are annotated with extra information which is a 'name' for the bound atom. See Theorem 8.20.

[11] There are no atoms-abstractions without finite support because in Definition 3.8 we only form $[a]x$ when x has finite support.

3.3. Concretion $\hat{x}@a$; the destructor for $[a]x$.

LEMMA 3.16. *Let $\hat{x}$ be an atoms-abstraction (with finite support) and let a be an atom such that $a\#\hat{x}$. Then there is a unique x such that $\hat{x} = [a]x$.*
Define $\hat{x}@a$ to be this unique element if it exists, and undefined otherwise.
Call $\hat{x}@a$ the concretion *of $\hat{x}$ at a.*

PROOF. Suppose $\hat{x} = [a]x'$. By construction $(a, x') \in [a]x'$. Also by construction, $x' = \pi\cdot x$ for $\pi \in \mathit{fix}(\mathit{supp}(x)\setminus\{a\})$, and it follows that $\pi(a) = a$. By Theorem 2.21 $\pi\cdot x = x$ and the result follows. ⊣

LEMMA 3.17. *$\hat{x}@a$ is defined if and only if $a\#\hat{x}$.*

PROOF. The right-to-left implication is Lemma 3.16. The left-to-right implication is from Theorem 3.11. ⊣

Corollary 3.18 states a formal sets sense in which an atoms-abstraction $\hat{x}$ is made up of its 'destructed parts', just as $(x, y) = \{\{x\}, \{x, y\}\}$ states how the pairset (x, y) is made up of its first and second projections x and y:

COROLLARY 3.18. $\hat{x} = \{(a, \hat{x}@a) \mid a\#\hat{x}\}$.

PROOF. By Lemma 3.16 $\hat{x}@a$ is the unique x such that $(a, x) \in \hat{x}$. The result follows by Lemma 3.17. ⊣

THEOREM 3.19. *Let x be an element with finite support. Let $\hat{x}$ be an element with finite support; by our convention in Remark* 3.9 *we assume that $\hat{x}$ is an atoms-abstracton.*

- *If $b\#x$ then $([a]x)@b$ is defined and $([a]x)@b = (b\ a)\cdot x$.*
- *$([a]x)@a = x$.*
- *If $a\#\hat{x}$ then $\hat{x}@a$ is defined and $[a](\hat{x}@a) = \hat{x}$.*

PROOF.
- Suppose $b\#x$. By Theorem 3.11 $b\#[a]x$. Combining Corollary 3.18 with Lemma 3.13 we deduce that $([a]x)@b = (b\ a)\cdot x$.
- As in the previous case.
- From Lemma 3.15 and Corollary 3.18. ⊣

§4. Meta-mathematical properties of $\mathcal{U}$. The proofs of properties of elements of $\mathcal{U}$ in this paper can be written out in a formal language, described below. We will use this to formally state and prove properties of the assertions we can make about $\mathcal{U}$. These are *meta-mathematical* properties; properties of properties of $\mathcal{U}$. The most important ones are rank induction (Theorem 4.2) and equivariance (Theorem 4.4, and its corollaries Corollary 4.6 and Theorem 4.7).

Consider first-order logic with:

- A binary predicate symbol $=$ called *equality*.
- A binary predicate symbol $\in$ called *set membership*.
- A constant term-former $\mathbb{A}$ called *the set of atoms*.

So, terms h and predicates ϕ are inductively defined by:

$$h ::= x, y, z, \cdots \mid \mathbb{A}$$
$$\phi ::= \bot \mid \phi \Rightarrow \phi \mid \forall x.\phi \mid h \in h \mid h = h$$

This is the language of ZFA set theory (Section 10) and is a sufficent language for much mathematics. For example, we can use this language to formally reason about $\mathcal{U}$ from Definition 2.4.

4.1. Rank induction. A notational convention will be very useful in what follows:

DEFINITION 4.1. We abbreviate a list of variables, such as $z_1, \ldots, z_n$, to $\overline{z}$. For example, $\phi(\overline{z}, x)$ is shorthand for $\phi(z_1, \ldots, z_n, x)$.

THEOREM 4.2. *Suppose that $\phi(\overline{z}, x)$ is a predicate mentioning variables included in $\overline{z}, x$. We consider $\overline{z}$ to be 'parameters' and x to be 'varying'. Then we have the following principle of* rank induction:

If

$$\forall y.(\forall x.rank(x) < rank(y) \Rightarrow \phi(\overline{z}, x)) \Rightarrow \phi(\overline{z}, y)$$

then

$$\forall x.\phi(\overline{z}, x).$$

PROOF. By contradiction. Suppose x denotes an element of least rank such that $\phi(\overline{z}, x)$ does not hold. So for all x' with $rank(x') < rank(x)$ it is the case that $\phi(\overline{z}, x')$. By assumption it follows that $\phi(\overline{z}, x)$. ⊣

4.2. Equivariance.

DEFINITION 4.3. Write

$$\pi{\cdot}\overline{x} \text{ for } \pi{\cdot}x_1, \ldots, \pi{\cdot}x_n.$$

THEOREM 4.4. *Suppose $\phi(\overline{x})$ is a predicate on some list of variables $\overline{x}$. Then the* principle of equivariance *states that*

$$\phi(\overline{x}) \Leftrightarrow \phi(\pi{\cdot}\overline{x}).$$

Note that $\overline{x}$ must contain *all* the variables mentioned in the predicate. Clearly, it is not the case that $a = a$ if and only if $a = b$—but it is the case that $a = b$ if and only if $b = a$.

PROOF. We work by induction on the syntax of ϕ.

- $x_i \in x_j$ implies $\pi{\cdot}x_i \in \pi{\cdot}x_j$ direct from Definition 2.14. The reverse implication is easy using π^{-1}.
- $x_i = x_j$ if and only if $\pi{\cdot}x_i = \pi{\cdot}x_j$ is also direct from Definition 2.14.
- The case of $\bot$ is trivial, and the cases of $\phi_1 \Rightarrow \phi_2$ and $\forall z.\phi'$ follow using the inductive hypothesis.
- $\pi{\cdot}\mathbb{A} = \mathbb{A}$, so $x_i \in \mathbb{A}$ if and only if $\pi{\cdot}x_i \in \mathbb{A}$, and $\mathbb{A} = x_j$ if and only if $\mathbb{A} \in \pi{\cdot}x_j$, and similarly $x_i = \mathbb{A}$ if and only if $\pi{\cdot}x_i = \mathbb{A}$ and $\mathbb{A} = x_j$ if and only if $\mathbb{A} = \pi{\cdot}x_j$.

The result follows. ⊣

DEFINITION 4.5. As is standard, we can specify a map χ using a predicate $\phi(\overline{x}, z)$ such that

$$\forall \overline{x}.\Big((\exists z.\phi(\overline{x}, z)) \wedge (\forall z, z'.\phi(\overline{x}, z) \wedge \phi(\overline{x}, z') \Rightarrow z = z')\Big).$$

COROLLARY 4.6. *Suppose $\chi(\overline{x})$ is a function specified using a list of variables $\overline{x}$. Then*

$$\boxed{\pi{\cdot}\chi(\overline{x}) = \chi(\pi{\cdot}\overline{x}).}$$

PROOF. We unpack Definition 4.5 and use equivariance (Theorem 4.4). ⊣

THEOREM 4.7. *Suppose $\chi(\overline{x})$ is a function on variables included in $\overline{x}$, which is $x_1, \dots, x_n$. Suppose $\overline{x}$ denotes elements with finite support. Then*

$$supp\,(\chi(\overline{x})) \subseteq supp\,(x_1) \cup \cdots \cup supp\,(x_n).$$

As a corollary, if χ is injective then

$$supp\,(\chi(\overline{x})) = supp\,(x_1) \cup \cdots \cup supp\,(x_n).$$

PROOF. The corollary follows by considering the result for χ and its inverse. Suppose that $\pi \in fix(supp\,(x_1) \cup \cdots \cup supp\,(x_n))$. We reason as follows:

$$\begin{aligned} \pi{\cdot}\chi(\overline{x}) &= \chi(\pi{\cdot}\overline{x}) && \text{Corollary 4.6} \\ &= \chi(\overline{x}) && \text{Theorem 2.21} \end{aligned}$$

The result follows. ⊣

LEMMA 4.8. *Suppose $\chi(\overline{x})$ is a function on variables included in $\overline{x}$, which is $x_1, \dots, x_n$. It is not necessarily the case that*

$$supp\,(\chi(\overline{x})) = supp\,(x_1) \cup \cdots \cup supp\,(x_n).$$

PROOF. It suffices to give a counterexample. We give two:

- $\chi(x) = \varnothing$ always.
- $\chi(x, y) = x \cup y$; take $x = \mathbb{A}\setminus\{a\}$ and $y = \{a\}$. So $supp\,(x) = \{a\} = supp\,(y)$ but $supp\,(x \cup y) = \varnothing$. ⊣

§5. Case study: inductively-defined λ-terms using atoms-abstraction. We are now ready to use atoms-abstraction to build datatypes of syntax-up-to-binding in nominal style; nominal abstract syntax. Λ and Λ_{nc} are two sets, both representing λ-calculus syntax. Λ_{nc} represents terms not up to α-equivalence—we might call this 'raw syntax'. In this section we shall make formal a natural sense in which Λ represents λ-calculus syntax up to α-equivalence (this is Theorem 5.18 and Corollary 5.19). Of course, we can build a set of terms up to α-equivalence just by quotienting Λ_{nc} by α-equivalence (Definition 5.17). However, Λ is inductively defined. In Section 7 we will exploit this, to develop new inductive reasoning principles on Λ.

First, a technical lemma:

LEMMA 5.1. *$a = (x, y)$, $a = [a]z$, $a = [b]z$, and $(x, y) = [a]z$ are all impossible.*

PROOF. By construction a has no members and (x, y) has one or two. Meanwhile, Lemma 3.13 makes it evident that $[a]z$ always has infinitely many members. ⊣

We propose an inductive datatype of untyped λ-calculus syntax using atoms-abstraction.

DEFINITION 5.2. Let Λ be the collection characterised by:[12]

$$\frac{a \in \mathbb{A}}{a \in \Lambda} \qquad \frac{t \in \Lambda \quad t' \in \Lambda}{tt' \in \Lambda} \qquad \frac{a \in \mathbb{A} \quad t \in \Lambda \quad t \text{ finitely supported}}{\lambda[a]t \in \Lambda}$$

Here, we write $t't$ for (t', t) and $\lambda[a]t$ for $[a]t$.

By Lemma 5.1 the three ways of constructing elements of Λ are mutually exclusive.

We define an inductive datatype of untyped λ-calculus syntax (not up to α-equivalence) using a more traditional *name-carrying* construction:

DEFINITION 5.3. Let Λ_{nc} be the collection characterised by:

$$\frac{a \in \mathbb{A}}{a \in \Lambda_{nc}} \qquad \frac{s \in \Lambda_{nc} \quad s' \in \Lambda_{nc}}{ss' \in \Lambda_{nc}} \qquad \frac{a \in \mathbb{A} \quad s \in \Lambda_{nc}}{\lambda a.s \in \Lambda_{nc}}$$

Here we write $\lambda a.s$ for $(\varnothing, (a, s))$.

REMARK 5.4. For the rest of this section, t, t', and t'' range over elements of Λ, and s, s', and s'' range over elements of Λ_{nc}. For example, 't' means 'some element of Λ' and '$\forall t.\phi(t)$' means 'for all t *in* Λ, $\phi(t)$'. ◇

[12]In fact, Λ is an element of $\mathcal{U}$, but we prove that in Lemma 5.7. First, we have to specify the extension of Λ—the elements it contains—and while we do this we call Λ a *collection*, as in 'collection of elements'.

Some bounds on the rank of elements will be useful:

LEMMA 5.5. • $rank((x, y)) = max(rank(x), rank(y)) + 2$.
- $rank(inl(x)) \leq rank(x) + 4$. $rank(inr(y)) \leq rank(y) + 4$.
- $rank(x) = rank(\pi{\cdot}x)$.
- $rank(u\circlearrowleft_A) = rank(u) + 1$.
- $rank([a]x) \leq rank(x) + 3$.

PROOF. • By Definition 2.7 $(x, y) = \{\{x\}, \{x, y\}\}$. Then

$$rank(\{x\}) = rank(x) + 1 \text{ and } rank(\{x, y\}) = max(rank(x), rank(y)) + 1,$$

and so

$$rank((x, y)) = max(rank(x), rank(y)) + 2.$$

- By Definition 2.9 $inl(x) = (\varnothing, x)$. It follows by the first part of this result that $rank(inl(x)) = max(1, rank(x)) + 2$.
  Also, $inr(y) = (\{\varnothing\}, y)$, and it follows that
  $$rank(inr(x)) = max(2, rank(x)) + 2.$$
  The result follows by arithmetic.
- $rank(\pi{\cdot}x) = rank(x)$ by Corollary 4.6.
- $rank(u\circlearrowleft_A) = rank(u) + 1$ follows by construction and since $rank(\pi{\cdot}x) = rank(x)$.
- $rank([a]x) \leq rank((a, x)) + 1$ follows by the previous parts and since $rank(a) = 0$. ⊣

By rank induction (Theorem 4.2) and Lemma 5.5 we obtain inductive principles:

THEOREM 5.6.

$$\boxed{\textit{If}\quad \begin{array}{ll} \forall a. & \phi(\overline{z}, a) \\ \forall t, t'. & (\phi(\overline{z}, t) \wedge \phi(\overline{z}, t')) \Rightarrow \phi(\overline{z}, tt') \\ \forall a.\forall t. & \phi(\overline{z}, t) \Rightarrow \phi(\overline{z}, \lambda[a]t) \end{array} \quad \textit{then} \quad \forall t.\phi(\overline{z}, t).}$$

Above, a ranges over atoms and t and t' range over elements of Λ.

$$\boxed{\textit{If}\quad \begin{array}{ll} \forall a. & \phi(\overline{z}, a) \\ \forall s, s'. & (\phi(\overline{z}, s) \wedge \phi(\overline{z}, s')) \Rightarrow \phi(\overline{z}, ss') \\ \forall a.\forall s. & \phi(\overline{z}, s) \Rightarrow \phi(\overline{z}, \lambda a.s) \end{array} \quad \textit{then} \quad \forall s.\phi(\overline{z}, s).}$$

Above, a ranges over atoms and s and s' range over elements of Λ_{nc}.

LEMMA 5.7. *Λ and Λ_{nc} are elements.*
In symbols: $\Lambda \in \mathcal{U}$ and $\Lambda_{nc} \in \mathcal{U}$.

PROOF. By a routine rank induction (Theorem 4.2) using Lemma 5.5 we prove that

$$\forall x.x \in \Lambda \Rightarrow rank(x) < \omega.$$

It follows by construction (Definition 2.4) that $\Lambda \in \mathcal{U}_\omega$ and so Λ is an element. The case of Λ_{nc} is similar. $\dashv$

An easy observation:

LEMMA 5.8. *The permutation action on elements in Λ and Λ_{nc} satisfies*:

$$\pi{\cdot}a = \pi(a), \qquad \pi(tt') = (\pi{\cdot}t)(\pi{\cdot}t'), \qquad \pi{\cdot}\lambda[a]t = \lambda[\pi(a)]\pi{\cdot}t,$$
$$\pi{\cdot}a = \pi(a), \qquad \pi(ss') = (\pi{\cdot}s)(\pi{\cdot}s'), \qquad \pi{\cdot}\lambda a.s = \lambda(\pi(a)).\pi{\cdot}s.$$

All elements of Λ and Λ_{nc} are finitely-supported, and the support of elements of Λ and Λ_{nc} satisfies:

$$supp\,(a)=\{a\},\;\; supp\,(tt')=supp\,(t)\cup supp\,(t'),\;\; supp\,(\lambda[a]t)=supp\,(t)\setminus\{a\},$$
$$supp\,(a)=\{a\},\;\; supp\,(ss')=supp\,(s)\cup supp\,(s'),\;\; supp\,(\lambda a.s)=supp\,(s)\cup\{a\}.$$

PROOF. The first part is Theorem 4.4 (or by an easy calculation). The second part follows by Theorem 5.6, Corollary 2.30, and Theorem 3.11. $\dashv$

REMARK 5.9. Support matches the usual notion of 'free variables of' for $t \in \Lambda$, but not for $s \in \Lambda_{nc}$. Elements of Λ_{nc} do not 'know' that λ is supposed to be a binder. Therefore we must define the notion of *free variables* separately, as in Definition 5.10. $\diamond$

DEFINITION 5.10. As is standard, we define the *free variables* $fv(s)$ inductively by:

$$fv(a) = \{a\}, \qquad fv(ss') = fv(s) \cup fv(s'), \qquad fv(\lambda a.s) = fv(s) \setminus \{a\}.$$

DEFINITION 5.11. Define a translation $\langle\text{-}\rangle$ inductively from Λ_{nc} to Λ by:

$$\langle a\rangle = a, \qquad \langle ss'\rangle = \langle s\rangle\langle s'\rangle, \qquad \langle\lambda a.s\rangle = \lambda[a]\langle s\rangle.$$

Support in Λ in the sense of Definition 2.16 corresponds with 'free variables' in Λ_{nc} under the translation of Definition 5.11:

LEMMA 5.12. $supp\,(\langle s\rangle) = fv(s)$.

PROOF. By an easy induction on s using Definition 5.10 and Lemma 5.8. $\dashv$

DEFINITION 5.13. For each finite set of atoms S make some fixed but arbitrary choice of 'fresh' atom c such that $c \notin S$.[13] We specify a *substitution* action on the name-carrying datatype Λ_{nc} by:

$$a[a \mapsto s''] = s'',$$
$$b[a \mapsto s''] = b,$$
$$(ss')[a \mapsto s''] = (s[a \mapsto s''])(s'[a \mapsto s'']),$$

[13] Note that this choice can be represented as an element in $\mathcal{U}$. Such a representation will not have finite support, but $\mathcal{U}$ is allowed to contain elements that do not have finite support because it is the cumulative hierarchy model of Zermelo–Fraenkel set theory with atoms (not of Fraenkel–Mostowski set theory). See Section 10.

$$(\lambda a.s)[a \mapsto s''] = \lambda a.s,$$
$$(\lambda b.s)[a \mapsto s''] = \lambda c.(s[b \mapsto c][a \mapsto s'']) \qquad (b \in fv(s''),\ c \text{ fresh}),$$
$$(\lambda b.s)[a \mapsto s''] = \lambda b.(s[a \mapsto s'']) \qquad (b \notin fv(s'')).$$

Here, c is chosen according to our fixed but arbitrary choice, fresh for a, b, and the atoms in s and s''.

Definition 5.13 is not inductive on syntax because in the clause for $(\lambda b.s)[a \mapsto s'']$, $s[b \mapsto c]$ is not a subterm of $\lambda b.s$. In fact, what we have is an inductively-specified relation, which we might write as "-$[a \mapsto s'']$ is related with -". For example, if s and s''' are related, and s' and s'''' are related, then so are $s(s')$ and $(s''')(s'''')$. We should prove that this specification does indeed specify a function:

Lemma 5.14. *Definition* 5.13 *specifies a well-defined function. Also,* $rank(s[b \mapsto c]) = rank(s)$.

Proof. By induction on $rank(s)$. ⊣

Definition 5.15 and Lemma 5.16 are key technical constructions with which the rest of the proofs (including that of Theorem 5.18) run quite smoothly:

Definition 5.15. Define the *bound atoms* $ba(s)$ inductively on Λ_{nc} by:

$$ba(a) = \varnothing, \qquad ba(ss') = ba(s) \cup ba(s'), \qquad ba(\lambda a.s) = ba(s) \cup \{a\}.$$

Lemma 5.16. *Suppose* $s \in \Lambda_{nc}$. *For all atoms* a *and* b, *if* b *does not occur in* s *(that is, if* $b\#s$*) and if* $a \notin ba(s)$, *then* $s[a \mapsto b] = (b\ a)\cdot s$.

Proof. By induction on s:

- The case of a. $a[a \mapsto b] = b = (b\ a)a$.
- The case of b. There is nothing to prove since b occurs in b.
- The case of c. $c[a \mapsto b] = c = (b\ a)c$.
- The case of ss'. $(ss')[a \mapsto b] = (s[a \mapsto b])(s'[a \mapsto b])$ and $(b\ a)\cdot(ss') = ((b\ a)\cdot s)((b\ a)\cdot s')$. We use the inductive hypothesis.
- The case of $\lambda a.s$. There is nothing to prove since $a \in ba(s)$.
- The case of $\lambda b.s$. There is nothing to prove since b occurs in $\lambda b.s$.
- The case of $\lambda c.s$. $(\lambda c.s)[a \mapsto b] = \lambda c.(s[a \mapsto b])$ and $(b\ a)\cdot(\lambda c.s) = \lambda c.(b\ a)\cdot s$. We use the inductive hypothesis. ⊣

Definition 5.17. Define *α-equivalence* $=_\alpha$ on Λ_{nc} to be the least equivalence relation such that:

$$\frac{}{a =_\alpha a} \qquad \frac{s =_\alpha s' \quad t =_\alpha t'}{st =_\alpha s't'} \qquad \frac{s =_\alpha s'}{\lambda a.s =_\alpha \lambda a.s'}$$

$$\frac{s[a \mapsto c] =_\alpha s'[b \mapsto c]}{\lambda a.s =_\alpha \lambda b.s'} \quad (c \text{ fresh}, a \notin ba(s), b \notin ba(s'))$$

Here, c is chosen according to the fixed but arbitrary choice in Definition 5.13, fresh for a, b, and the atoms in s and s'.

THEOREM 5.18. *$s =_\alpha s'$ if and only if $\langle s\rangle = \langle s'\rangle$.*

PROOF. We prove two implications.

We prove by induction on s that

$$\text{for all } s', \text{ if } s =_\alpha s' \text{ then } \langle s\rangle = \langle s'\rangle.$$

We consider only one case. Suppose

$$s[a \mapsto c] =_\alpha s'[b \mapsto c]$$

where c is fresh and $a \notin ba(s)$ and $b \notin ba(s')$. By Lemma 5.16 $(c\ a)\cdot s = (c\ b)\cdot s'$, so also $\langle (c\ a)\cdot s\rangle = \langle (c\ b)\cdot s'\rangle$. By Theorem 4.4 $(c\ a)\cdot\langle s\rangle = (c\ b)\cdot\langle s'\rangle$. By Lemma 5.12 $c\#\langle s\rangle$ and $c\#\langle s'\rangle$. By part 2 of Lemma 3.12 $[a]\langle s\rangle = [b]\langle s'\rangle$ so

$$\langle \lambda a.s\rangle = \lambda[a]\langle s\rangle = \lambda[b]\langle s'\rangle = \langle \lambda b.s'\rangle.$$

We also prove by induction on $rank(s)$ that

$$\text{for all } s', \text{ if } \langle s\rangle = \langle s'\rangle \text{ then } s =_\alpha s'.$$

We consider only one case. Suppose that $\lambda[a]\langle s\rangle = \lambda[b]\langle s'\rangle$. By suitable α-conversions in s and s' we can ensure that $a \notin ba(s)$ and $b \notin ba(s')$. By equivariance these have the same rank as the originals so we retain the inductive hypothesis. So assume without loss of generality that $a \notin ba(s)$ and $b \notin ba(s')$. Let c be the choice of fresh atom for the atoms in s, s', a, and b. By Lemma 5.8 $c\#s$ and $c\#s'$, so by Theorem 4.7 $c\#\langle s\rangle$ and $c\#\langle s'\rangle$. Therefore by part 2 of Lemma 3.12

$$(c\ a)\cdot\langle s\rangle = (c\ b)\cdot\langle s'\rangle.$$

By Theorem 4.4 and Lemma 5.16

$$(c\ a)\cdot\langle s\rangle = \langle s[a \mapsto c]\rangle \text{ and } (c\ b)\cdot\langle s'\rangle = \langle s'[b \mapsto c]\rangle.$$

It follows by the inductive hypothesis that $s[a \mapsto c] =_\alpha s'[b \mapsto c]$. We conclude that $\lambda a.s =_\alpha \lambda b.s'$ as required. ⊣

COROLLARY 5.19. *$\langle\text{-}\rangle$ expresses a bijection between Λ and λ-terms up to α-equivalence.*

PROOF. In view of Theorem 5.18 it suffices to prove surjectivity; for every $t \in \Lambda$ there exists some $s \in \Lambda_{nc}$ such that $t = \langle s\rangle$. This is routine by induction on $rank(t)$. ⊣

The next step is to explore 'nominal' inductive reasoning principles for Λ. We do this in Section 7, after we have constructed the И-quantifier.

§6. The И quantifier and the 'fresh' binder.

6.1. The И quantifier. The И-quantifier is a characteristic feature of nominal techniques, and was introduced in [37]. $\text{И}a.\phi$ holds when ϕ holds of all but finitely many atoms (see Definition 6.1 for the formal definition). That is, И means 'all but finitely many'.

We can make connections here with quantifiers for 'most' and 'many' [69] and with *generalised quantifiers* [44, Section 1.2.1], which are of particular interest in linguistics. Cheney generalised the notion of 'finitely many' using ultrafilters, in the study of completeness for nominal logic [10], and the author has considered a generalisation of 'finite' to 'well-orderable', in the study of elements with infinite support and with abstraction by infinitely many atoms [26].

The notion of 'all but finitely many' is not new to the study of variables in syntax. Krivine used the notion in his book [45, Section 2, 'α-equivalence and substitution']. What makes the И in this paper different and useful is the way it interacts with equivariance and finite support; these give the И-quantifier its characteristic some/any property (Theorem 6.5) and commutations with connectives and quantifiers (Theorem 6.6).

Using support, И can be expressed using $\forall$, and also using $\exists$ (Theorem 6.5). A related property of И is *self-duality*—when the left- and right-introduction rules of a quantifier are symmetric (Remark 6.7).[14]

We will use И in this paper, beginning with its application to inductive reasoning principles on syntax with binding in Section 7.

Definition 6.1. Suppose $\phi(\overline{z}, a)$ is a predicate on variables included in $\overline{z}, a$—here $\overline{z}$ is shorthand for 'any other variables mentioned in ϕ', and we intend a to range over atoms.

The *NEW quantifier* $\text{И}a.\phi(\overline{z}, a)$ is defined by

$$\boxed{\text{И}a.\phi(\overline{z}, a) \text{ is true when } \{a \in \mathbb{A} \mid \phi(\overline{z}, a) \text{ is false}\} \text{ is finite.}}$$

Freshness $a\#x$ (Definition 2.16) can be characterised directly using И and equality (see [37, Equation 5] or [37, Equation 13]):

Theorem 6.2. *Let x be an element with finite support. Then*

$$\boxed{a\#x \textit{ if and only if } \text{И}b.(b\ a)\cdot x = x.}$$

Proof. Suppose $a\#x$. By Theorem 2.21 if $(b\ a)\cdot x \neq x$ then $b \in supp\,(x)$. $supp\,(x)$ is finite by assumption. The result follows.

[14] The proof theory of the И quantifier was considered by the author [25] and in collaboration with Cheney [29], and then improved by Cheney [9]. Another view on И is given in [16, Subsection 7.3].

Now suppose that $\text{И}b.(b\ a)\cdot x = x$. Choose any pair of distinct atoms b and b' such that $(b\ a)\cdot x = x$ and $(b'\ a)\cdot x = x$. Note that

$$(b\ b') = (b\ b') \circ (b\ a) \circ (b\ a) = (b\ a) \circ (b'\ a) \circ (b\ a).$$

Therefore $(b\ b')\cdot x = x$ always. It follows that if $\pi \in \mathit{fix}(B \cup \{a\})$ then $\pi\cdot x = x$. Therefore $a \notin \mathit{supp}\,(x)$. ⊣

Corollary 6.3. *x has finite support if and only if* $\text{И}a.\text{И}b.(b\ a)\cdot x = x$.

Proof. Suppose x has a finite supporting set $S \subseteq \mathbb{A}$. By Theorem 2.21 if $a, b \notin S$ then $(b\ a)\cdot x = x$. It follows that $\text{И}a.\text{И}b.(b\ a)\cdot x = x$.

Conversely suppose that $\text{И}a.\text{И}b.(b\ a)\cdot x = x$. By Theorem 6.2 $\text{И}a.a\#x$. Unpacking Definitions 2.16 and 6.1 this tells us that $\{a \mid a \in \mathit{supp}\,(x)\} = \mathit{supp}\,(x)$ is finite. By Theorem 2.21 $\mathit{supp}\,(x)$ is a finite supporting set for x. ⊣

Definition 6.4. If $\overline{z}$ is a list of variables $z_1, \dots, z_n$ write

$$a\#\overline{z} \text{ for } a\#z_1 \wedge \cdots \wedge a\#z_n.$$

Theorem 6.5 expresses the characteristic some/any property of the И-quantifier:

Theorem 6.5. *Suppose $\phi(\overline{z}, a)$ is a predicate on variables included in $\overline{z}, a$. Suppose $\overline{z}$ denotes a list of elements with finite support. Then the following are equivalent*:

$\forall a.\ (a \in \mathbb{A} \wedge a\#\overline{z}) \Longrightarrow \phi(\overline{z}, a)$	$\forall$ *form of* $\text{И}a.\phi(\overline{z}, a)$
$\text{И}a.\phi(\overline{z}, a)$	
$\exists a.\ a \in \mathbb{A} \wedge a\#\overline{z} \wedge \phi(\overline{z}, a)$	$\exists$ *form of* $\text{И}a.\phi(\overline{z}, a)$

Proof. All top-to-bottom implications are easy. Now suppose there exists some atom a such that

$$a\#\overline{z} \wedge \phi(\overline{z}, a).$$

Choose any other atom b such that $b\#\overline{z}$. By Theorems 4.4 and 2.21 it follows that $\phi(\overline{z}, b)$. The result follows. ⊣

Theorem 6.6. *Suppose $\overline{z}$ denotes a list of elements with finite support.*

1. *Suppose $\phi(\overline{z})$ and $\psi(\overline{z}, a)$ are predicates on variables included in $\overline{z}$ and $\overline{z}, a$ respectively. If $a\#\overline{z}$ then*

$$\boxed{\phi(\overline{z})\ \mathit{op}\ \text{И}a.\psi(\overline{z}, a) \Leftrightarrow \text{И}a.(\phi(\overline{z})\ \mathit{op}\ \psi(\overline{z}, a)).}$$

   Here, op ranges over $\wedge$, $\vee$, and $\Rightarrow$.
2. *Suppose $\phi(\overline{z}, a, x)$ is a predicate on variables included in $\overline{z}, a, x$. Write $\mathit{fs}(x)$ for the assertion ‘x is an element of finite support’, formalised in*

the language of ZFA (for instance as in Definition 2.16 *or Corollary* 6.3). *Then*:

$$\boxed{\begin{array}{ll} \forall x.fs(x) \Rightarrow \mathsf{N}a.\phi(\overline{z},a,x) & \Leftrightarrow \mathsf{N}a.\forall x.(fs(x)\wedge a\#x) \Rightarrow \phi(\overline{z},a,x) \\ \exists x.fs(x) \wedge \mathsf{N}a.\phi(\overline{z},a,x) & \Leftrightarrow \mathsf{N}a.\exists x.(fs(x)\wedge a\#x) \wedge \phi(\overline{z},a,x) \end{array}}$$

3. *Suppose* $\phi(\overline{z},a)$ *and* $\psi(\overline{z},a)$ *are predicates on variables included in* $\overline{z},a$. *Then*

$$\boxed{\begin{array}{c} \mathsf{N}a.(\phi(\overline{z},a)\ op\ \psi(\overline{z},a)) \Leftrightarrow (\mathsf{N}a.\phi(\overline{z},a))\ op\ (\mathsf{N}a.\psi(\overline{z},a)) \\ \mathsf{N}a.\neg\phi(\overline{z},a) \Leftrightarrow \neg\mathsf{N}a.\phi(\overline{z},a). \end{array}}$$

Here, op ranges over $\wedge$, $\vee$, *and* $\Rightarrow$.

PROOF. 1. Using Theorem 6.5 and elementary properties of $\forall$ and $\exists$.
2. Easy using Theorem 6.5.
3. We consider the case of implication, which is the least trivial.

Suppose $\mathsf{N}a.(\phi(\overline{z},a) \Rightarrow \psi(\overline{z},a))$ and $\mathsf{N}a.\phi(\overline{z},a)$. By Theorem 6.5 if $a\#\overline{z}$ then $\phi(\overline{z},a) \Rightarrow \psi(\overline{z},a)$ and $\phi(\overline{z},a)$. Choose some $a\#\overline{z}$; it follows that $\psi(\overline{z},a)$ so by Theorem 6.5 we conclude that $\mathsf{N}a.\psi(\overline{z},a)$.

Now suppose $(\mathsf{N}a.\phi(\overline{z},a)) \Rightarrow (\mathsf{N}a.\psi(\overline{z},a))$. We use Theorem 6.5; it suffices to choose $a\#\overline{z}$ and prove $\phi(\overline{z},a) \Rightarrow \psi(\overline{z},a)$. Suppose $\phi(\overline{z},a)$. By Theorem 6.5 $\mathsf{N}a.\phi(\overline{z},a)$, therefore $\mathsf{N}a.\psi(\overline{z},a)$, and by Theorem 6.5 we deduce $\psi(\overline{z},a)$ as required. ⊣

REMARK 6.7. We call $\neg\mathsf{N}a.\phi \Leftrightarrow \mathsf{N}a.\neg\phi$ *self-duality*. In fact, self-duality is when the left- and right-introduction rules are symmetric with each other; in the case of classical logic with two truth-values, this is equivalent to commuting with classical negation.

Miller and Tiu have developed another self-dual quantifier ∇, with quite similar properties [64, 51]. Tiu's recent work [64] also emphasises equivariance. ◇

6.2. The 'fresh' binder. Intuitively, N generates fresh atoms in predicates. Sometimes it is useful to generate fresh atoms in functions too.

DEFINITION 6.8. Suppose $\overline{z}$ denotes a list of elements with finite support. Suppose $\chi(\overline{z},a)$ is a function specified using $\overline{z}$ and a variable a considered to range over atoms, and suppose

$$\mathsf{N}a.a\#\chi(\overline{z},a).$$

Specify $\mathsf{fr}\,a.\chi(\overline{z},a)$ by

$$\boxed{\mathsf{N}b.(\mathsf{fr}\,a.\chi(\overline{z},a)) = \chi(\overline{z},b).}$$

LEMMA 6.9. *Definition* 6.8 *is well-defined, and* $\mathsf{fr}\,a.\chi(\overline{z},a) = \chi(\overline{z},b)$ *for any fresh* b *(so* $b\#\overline{z}$*).*

PROOF. By Theorem 6.5 ($\exists$ form), it suffices to show that if $a\#\overline{z}$ and $b\#\overline{z}$ then $\chi(\overline{z},a)=\chi(\overline{z},b)$.

By Theorem 6.5 ($\forall$ form) we know $a\#\chi(\overline{z},a)$ and $b\#\chi(\overline{z},b)$. By Theorem 2.21 $(b\ a)\cdot\chi(\overline{z},a)=\chi(\overline{z},a)$. By Corollary 4.6 $(b\ a)\cdot\chi(\overline{z},a)=\chi((b\ a)\cdot\overline{z},b)$ (here $(b\ a)\cdot\overline{z}$ is the vector of $(b\ a)\cdot z_i$ for each z_i in $\overline{z}$). By Theorem 2.21 $(b\ a)\cdot\overline{z}=\overline{z}$. The result follows. ⊣

The 'fresh' binder satisfies algebraic properties analogous to those satisfied by И. These were useful when the author implemented nominal theory in Isabelle [23]. Nominal Isabelle, a major reimplementation and extension, was then developed and applied to significant case studies [66].

THEOREM 6.10. *Suppose $\overline{z}$ denotes a list of elements with finite support. Suppose $\chi(\overline{z},a)$ and $\xi(\overline{z},a)$ are functions specified using $\overline{z}$ and a variable a considered to range over atoms. Suppose $\eta(\overline{z},x,y)$ is a function specified using $\overline{z}$ and x and y.*

For clarity and by abuse of notation, we will just write $\chi(a)$, $\xi(a)$, and $\eta(x,y)$; we take $\overline{z}$ to be understood.

Suppose

$$\text{И}a.a\#\chi(a) \textit{ and } \text{И}a.a\#\xi(a).$$

Then

$$\boxed{\begin{aligned}&\eta(\mathsf{fr}\,a.\chi(a),\mathsf{fr}\,a.\xi(a))=\mathsf{fr}\,a.\eta(\chi(a),\xi(a)) \qquad \textit{and}\\ &\eta(\mathsf{fr}\,a.\chi(a),\mathsf{fr}\,a.\xi(a))=\mathsf{fr}\,a.\mathsf{fr}\,b.\eta(\chi(a),\xi(b)).\end{aligned}}$$

PROOF. We must show that $\mathsf{fr}\,a.(\eta(\chi(a),\xi(a))$ is well-defined. Choose fresh a (so that $a\#f,g,h,X,Y,Z$). It suffices to prove that $a\#\eta(\chi(a),\xi(a))$. This is easy, since $a\#\chi(a)$ and $a\#\xi(a)$ by Theorem 6.5 ($\forall$ form), and by Theorem 4.7.

To prove the equality, by Theorem 6.5 ($\exists$ form) it suffices to choose fresh a (so that $a\#f,g,h,X,Y,Z$) and show that

$$\eta(\chi(a),\xi(a))=\eta(\mathsf{fr}\,a.\chi(a),\mathsf{fr}\,a.\xi(a)).$$

By Theorem 6.5 ($\forall$ form) we know that $\mathsf{fr}\,a.\chi(a)=\chi(a)$ and $\mathsf{fr}\,a.\xi(a)=\xi(a)$. The result follows.

The proof of the second part is similar. ⊣

§7. Application to reasoning on abstract syntax with binding. Recall the inductive principle for Λ given in Theorem 5.6; it treats atoms-abstraction $[a]$- (Definition 3.8) like an ordered pair. It is true that $[a]x$ resembles the pair (a,x) (Definition 2.7), and $[a]$- and -@a (Lemma 3.16) behave very much like $(a,\text{-})$ and second projection. However, $[a]x$ satisfies some equalities that (a,x) does not, as laid out in Theorem 3.19. As a result:

- Λ is inductive, like Λ_{nc} and unlike Λ_{nc} quotiented by α-equivalence.
- Λ is in bijection with λ-terms up to α-equivalence, unlike Λ_{nc} and like Λ_{nc} quotiented by α-equivalence.

Theorem 7.1 exploits this to present an alternative inductive principle. This result is the motivating observation of [37] and of its implementations, starting with those in the author's thesis [23] and progressing, for example, to Nominal Isabelle package and FreshML [67, 66, 59].

THEOREM 7.1. *Suppose $\phi(\overline{z}, t)$ is a predicate.*

$$\boxed{\textit{If}\quad \begin{array}{ll} \forall a. & \phi(\overline{z}, a) \\ \forall t, t'. & \phi(\overline{z}, t) \wedge \phi(\overline{z}, t') \Rightarrow \phi(\overline{z}, tt') \\ \mathsf{N}a.\forall t. & \phi(\overline{z}, t) \Rightarrow \phi(\overline{z}, \lambda[a]t) \end{array} \quad \textit{then} \quad \forall t.\phi(\overline{z}, t).}$$

Here a ranges over atoms and t and t' range over elements of Λ.

PROOF. Suppose $u \in \Lambda$ has least rank such that $\phi(\overline{z}, u)$ does not hold. We reason by cases to deduce a contradiction:

- Suppose $u = a$. By assumption $\phi(\overline{z}, u)$, a contradiction.
- Suppose $u = t't$. By Lemma 5.5 $\mathit{rank}(t') < \mathit{rank}(u)$ and $\mathit{rank}(t) < \mathit{rank}(u)$ so by inductive hypothesis $\phi(\overline{z}, t')$ and $\phi(\overline{z}, t)$, therefore by assumption $\phi(\overline{z}, u)$, a contradiction.
- Suppose $u = \lambda\hat{t}$ where $\hat{t}$ is an atoms-abstraction. Choose fresh a (so $a\#z$ and $a\#\hat{t}$). By part 3 of Theorem 3.19 $\hat{t}@a$ exists and $\hat{t} = [a](\hat{t}@a)$. By Lemma 5.5 $\mathit{rank}(\hat{t}@a) < \mathit{rank}(\lambda\hat{t})$ so by inductive hypothesis $\phi(\overline{z}, \hat{t}@a)$. By Theorem 6.5 ($\forall$ form) since $a\#z$ by assumption $\phi(\overline{z}, \lambda[a](\hat{t}@a))$, a contradiction. $\dashv$

We take a moment to state a variant of the result stated in Theorem 7.1, which uses concretion -@a instead of atoms-abstraction $[a]$-:

$$\boxed{\text{If}\quad \begin{array}{ll} \forall a. & \phi(\overline{z}, a) \\ \forall t, t'. & \phi(\overline{z}, t) \wedge \phi(\overline{z}, t') \Rightarrow \phi(\overline{z}, tt') \\ \forall \hat{t}.\mathsf{N}a. & \phi(\overline{z}, \hat{t}@a) \Rightarrow \phi(\overline{z}, \lambda\hat{t}) \end{array} \quad \text{then}\ \forall t.\phi(\overline{z}, t).}$$

Here a ranges over $\mathbb{A}$, t and t' range over elements of Λ, and $\hat{t}$ ranges over atoms-abstractions of elements in Λ (recall Remark 3.9). This variant may be more convenient if we think in terms of 'decomposing' a term, but this is purely a matter of convenience and the two forms are logically equivalent.

We can give a similar principle for defining functions out of Λ; this inductive principle is Theorem 6.5 of [37]. Treatments are also in [23, Subsection 10.4], and subsequently in [57]:

THEOREM 7.2. *Suppose we are given functions $F_{\mathsf{var}}(\overline{z}, x)$, $F_{\mathsf{app}}(\overline{z}, x', x)$, and $F_{\mathsf{lam}}(\overline{z}, x)$, from $\mathcal{U}$ to $\mathcal{U}$. For each fixed sequence of values for the variables $\overline{z}$,* *there exists a unique function-set f such that $dom(f) = \Lambda$ and such that*

$$\forall a.\ f(a) = F_{\mathsf{var}}(\overline{z}, a),$$
$$\forall t, t'.\ f(t't) = F_{\mathsf{app}}(\overline{z}, f(t'), f(t)),$$
$$\text{И}a.\forall t.\ f(\lambda[a]t) = F_{\mathsf{lam}}(\overline{z}, [a]f(t)).$$

Here a ranges over atoms and t and t' range over elements of Λ.

SKETCH PROOF. Let $\phi(\overline{z}, t)$ be the predicate

"if $t \in \Lambda$ then $f(t)$ exists and is uniquely defined"

formalised in the language of ZFA.[15]

We use the inductive principle described in Theorem 7.1:

- The cases of a and $t't$ are easy.
- We now prove that $\text{И}a.\forall t.\phi(\overline{z}, t) \Rightarrow \phi(\overline{z}, \lambda[a]t)$. By Theorem 6.5 ($\exists$ form) it suffices to choose a fresh (so $a\#z$), to choose any t, to suppose $\phi(\overline{z}, t)$ and then to prove $\phi(\overline{z}, \lambda[a]t)$. By Theorem 6.5 ($\forall$ form) since $a\#z$, $f(\lambda[a]t) = F_{\mathsf{lam}}(\overline{z}, [a]f(t))$. By assumption $f(t)$ exists and is uniquely defined, therefore so is $f(\lambda[a]t)$. The result follows. ⊣

REMARK 7.3. We can use part 2 of Theorem 6.6 and Theorem 3.19 to replace the bottom line of the inductive reasoning principle with the logically equivalent

$$\forall \hat{t}.\text{И}a.f(\lambda\hat{t}) = F_{\mathsf{lam}}(\overline{z}, [a]f(\hat{t}@a)).$$

Here $\hat{t}$ ranges over atoms-abstractions of elements of Λ (recall Remark 3.9). ◇

EXAMPLE 7.4. We use the recursive principle of Theorem 7.2 to define a *capture-avoiding substitution* $t'[a \mapsto t]$, where $t \in \Lambda$, $t' \in \Lambda$, and $a \in \mathbb{A}$. We select:

- $F_{\mathsf{var}}(a) = t$ and $F_{\mathsf{var}}(b) = b$ for any (other) atom b.
- $F_{\mathsf{app}}(t', t) = t't$.
- $F_{\mathsf{lam}}([c]t) = \lambda[c]t$.

Using Theorem 6.5 we unpack the recursive principle to:

$$a[a \mapsto t] = t,$$
$$b[a \mapsto t] = b,$$
$$(t''t')[a \mapsto t] = (t''[a \mapsto t])(t'[a \mapsto t]),$$
$$c\#t \Rightarrow (\lambda[c]t')[a \mapsto t] = \lambda[c](t'[a \mapsto t]).$$

Here $t'' \in \Lambda$.

[15] The formalisation is not entirely evident. We consider the set of all function-sets defined on initial segments of Λ and prove they must agree where they are defined. For more details see [42, Chapter 5], in particular Theorem 5.4.

This corresponds with the definition of capture-avoiding substitution as written out informally, following Lemma 5.12 and letting $c\#t$ correspond with 'c not free in t'.

§8. Type-formers: $[\mathbb{A}]X$ and $X_{\#a}$. For typed systems based on nominal techniques, it is important to understand the behaviour of sets of atoms-abstractions. Of particular interest is the interaction with sets union, cartesian product, disjoint sum, and function-spaces. Also of interest are functions out of sets of atoms-abstractions, since this structure influences the recursive, inductive, and pattern-matching principles which can be developed for programming destructors on types of atoms-abstractions. In this section we summarise, and generalise, what is known about sets of atoms-abstraction.

8.1. $[\mathbb{A}]X$ the abstraction type-former. Some terminology will be useful.

Definition 8.1. Suppose that X is a set. If X has finite support say that X has finite support to *level* 0. If X has finite support and all set members of X have finite support to level i, say that X has finite support to *level* $i+1$. We will only need finite support to levels 0 and 1.

We define the abstraction type-former $[\mathbb{A}]X$, following [37, Definition 5.4]:

Definition 8.2. Let X be a set with finite support to level 1.[16] Define

$$\boxed{[\mathbb{A}]X = \{[a]x \mid a \in \mathbb{A},\ x \in X,\ a\#X\}.}$$

A motivation for considering $[\mathbb{A}]X$ is to inductively model syntax-with-binding. The construction of inductive datatypes is based on fixedpoints for *monotone* functions.

Theorem 8.3. $[\mathbb{A}]$- *is monotone on sets with finite support to level* 1.
That is, if X and Y are sets with finite support to level 1 *then $X \subseteq Y$ implies* $[\mathbb{A}]X \subseteq [\mathbb{A}]Y$.

Proof. Suppose $\hat{x} \in [\mathbb{A}]X$. By construction there exists some $x \in X$ and $a\#X$ such that $\hat{x} = [a]x$. If $a\#Y$ then $\hat{x} \in [\mathbb{A}]Y$ and we are done. So suppose $a \in supp\,(Y)$. Choose fresh b (so $b\#\hat{x}, a, x, X, Y$). By Lemma 3.12 $\hat{x} = [b](b\ a)\cdot x$. By Theorems 4.4 and 2.21 $(b\ a)\cdot x \in X$, so also $(b\ a)\cdot x \in Y$. It follows that $\hat{x} \in [\mathbb{A}]Y$. ⊣

Definition 8.2 is oriented towards constructing elements of $[\mathbb{A}]X$. There is another characterisation of $[\mathbb{A}]X$ more oriented towards destructing elements:

[16] We require the elements $x \in X$ to have finite support so that $[a]x$ makes sense. This condition might be relaxed at the expense of complicating the definition. Seeing as we will not use the extra generality we retain the stricter, easier condition on X.

LEMMA 8.4. *Let X be a set with finite support to level* 1. *Then*

$$[\mathbb{A}]X = \{[a]x \mid a \in \mathbb{A},\ x \in X,\ \text{И}b.([a]x)@b \in X\}.$$

PROOF. We prove two set inclusions.

- $[\mathbb{A}]X \subseteq \{[a]x \mid a \in \mathbb{A},\ x \in X,\ \text{И}b.([a]x)@b \in X\}$.
  Suppose $a \in \mathbb{A}$, $x \in X$, and $a\#X$. We use Theorem 6.5 ($\exists$ form). Choose fresh b (so $b\#x, X$). By Theorems 4.4 and 2.21 $(b\ a)\cdot x \in X$. By part 1 of Theorem 3.19 $(b\ a)\cdot x = ([a]x)@b$. The result follows.
- $\{[a]x \mid a \in \mathbb{A},\ x \in X,\ \text{И}b.([a]x)@b \in X\} \subseteq [\mathbb{A}]X$.
  Suppose $a \in \mathbb{A}$, $x \in X$, and $\text{И}b.([a]x)@b \in X$. We use Theorem 6.5 ($\exists$ form). Choose fresh b (so $b\#x, X$). By part 3 of Theorem 3.19 $[b](([a]x)@b) = [a]x$. The result follows. ⊣

8.2. The set of finitely-supported functions $X \Rightarrow Y$. This short subsection is devoted to sets of finitely-supported functions $X \Rightarrow Y$ (Definition 8.5). This is used in Subsection 8.3 below, but it is more generally important than that. For example, $X \Rightarrow Y$ is used to build the arrows in the category FMSet in Definition 9.5, and it is the basis of the exponential of the categories FMSet and NOMSet (Lemma 9.7).

DEFINITION 8.5. Let X and Y be sets. Define

$$\boxed{X \Rightarrow Y = \{f \in X \to Y \mid f \text{ has finite support}\}.}$$

In words: $X \Rightarrow Y$ is the set of finitely-supported functions from X to Y.

REMARK 8.6. It is not the case that $(X \Rightarrow Y) = (X \to Y)$ in general. For example take $X = \mathbb{A}$ and $Y = \mathbb{B}$. Recall *comb* from Remark 2.18. Define

$$f = \bigcup \{(a, 1) \mid a \in comb\} \cup \{(a, 0) \mid a \in \mathbb{A}\backslash comb\}.$$

The reader can verify that $f \in (\mathbb{A} \to \mathbb{B}) \setminus (\mathbb{A} \Rightarrow \mathbb{B})$.

Note that $X \Rightarrow Y$ can be non-empty even if X and Y contain elements without finite support. For example, take X to be the set of total orderings of $\mathbb{A}$, and take $Y = X$. Then the identity function-set is in $X \Rightarrow Y$, as are the function-set for 'reverse the order', and many other functions that treat their arguments equivariantly.

Similarly, $X \Rightarrow Y$ can be non-empty even if Y does not have finite support. However, if X does not have finite support then no $f \in X \to Y$ has finite support and $X \Rightarrow Y$ is the empty set. We could address this, if we were so inclined, by considering a function space of finitely-supported *partial* functions. ◇

8.3. $X_{\#a}$ the a-fresh type-former.

DEFINITION 8.7. Suppose X and Y are sets and suppose $f \in X \to Y$ (Definition 2.12) is a function-set. Write $X \cong_f Y$ when f is a bijection. Write $X \cong Y$ when some f exists such that $X \cong_f Y$.

Definition 8.8 reprises a comment in [36, Section 7, page 9]. The mathematics that follows on from it is new:

DEFINITION 8.8. If X is a set with finite support to level 1 write

$$\boxed{X_{\#a} = \{x \in X \mid a\#x\}.}$$

Call this the *a-fresh* version of X.

If X and Y are sets with finite support to level 1 and $f \in X \to Y$, write

$$\boxed{f_{\#a} = \{(x, f(x)) \mid x \in X_{\#a}\}.}$$

We will be most interested in $X_{\#a}$ and $f_{\#a}$ for the case that $a\#X$ and $a\#f$.

LEMMA 8.9. *$a\#X$ does not necessarily imply that $X_{\#a} = X$.*

PROOF. Consider $a\#\mathbb{A}$. Then $\mathbb{A}_{\#a} = \mathbb{A}\backslash\{a\} \neq \mathbb{A}$. ⊣

A basic result of a-fresh sets is Theorem 8.10:

THEOREM 8.10. *Suppose X and Y are sets with finite support to level 1. Suppose $f \in X \to Y$. Suppose a is an atom such that $a\#X$, $a\#Y$, and $a\#f$. Then*

$$X \cong_f Y \text{ if and only if } X_{\#a} \cong_{f_{\#a}} Y_{\#a}.$$

As an immediate corollary, $X = Y$ if and only if $X_{\#a} = Y_{\#a}$.

Intuitively, this means that if X is a set and $a\#X$, then $X_{\#a}$ captures all of the important structure present in X.

PROOF. Suppose $X \cong_f Y$. It follows using Theorem 4.7 that $f_{\#a} \in X_{\#a} \to Y_{\#a}$. By a similar argument for the inverse of f, and by some elementary calculations, we see that $f_{\#a}$ is a bijection.

Now suppose $X_{\#a} \cong_{f_{\#a}} Y_{\#a}$. We must show that $f \in X \to Y$ is surjective and injective.

- *Surjective.* Choose any element $y \in Y$. If $a\#y$ then $y \in Y_{\#a}$ and by assumption $y = f(x)$ for some $x \in X_{\#a} \subseteq X$. So suppose $a \in supp(y)$. Let b be a fresh atom (so $b\#X, Y, f, a, y$). By Theorem 2.21 $(b\ a)\cdot Y = Y$. By equivariance $(b\ a)\cdot y \in Y$. By Theorem 2.19 $a\#(b\ a)\cdot y$, so $(b\ a)\cdot y = f(x)$ for some $x \in X_{\#a} \subseteq X$. By Theorem 2.21 $(b\ a)\cdot x = X$ and $(b\ a)\cdot f = f$. By equivariance $(b\ a)\cdot x \in X$ and $y \in f((b\ a)\cdot x)$.
- *Injective.* Choose any two distinct elements $x', x \in X$. Let b be a fresh atom (so that $b\#X, Y, f, a, x', x$). By Theorem 2.19 $a\#(b\ a)\cdot x'$ and $a\#(b\ a)\cdot x$. By Theorem 2.21 $(b\ a)\cdot x = X$ and $(b\ a)\cdot f = f$. By equivariance $(b\ a)\cdot x' \in X$ and $(b\ a)\cdot x \in X$. By assumption $f((b\ a)\cdot x') \neq f((b\ a)\cdot x)$. By equivariance again, $f(x') \neq f(x)$.

For the corollary, if $X = Y$ then $X_{\#a} = Y_{\#a}$ is immediate. Conversely, if $X_{\#a} = Y_{\#a}$ then we take f to be the identity on X, so that $f_{\#a}$ is the identity of $X_{\#a}$, and we deduce that $X \cong_f Y$, that is, $X = Y$. ⊣

We now prove that $\text{-}_{\#a}$ distributes over several constructs which are useful in constructing inductive (and indeed coinductive) datatypes.

Lemma 8.11. *Let X and Y be two sets with finite support to level* 1. *Let a be an atom such that $a\#X, Y$. Then*

$$(X \; op \; Y)_{\#a} = X_{\#a} \; op \; Y_{\#a}$$

for op one of $\times, +, \cup$. *In addition, if f is a function-set then*

$$\Big(\bigcup_{i\in dom(f)} f(i)\Big)_{\#a} = \bigcup_{i\in dom(f)} f(i)_{\#a}.$$

Proof. The cases of $\times$, $+$, and $\cup$ are by easy calculations using Corollary 2.30. The case of unions is routine. ⊣

Recall the definition of the 'fresh' binder $\mathsf{fr}\, a.\chi$ from Definition 6.8.

Lemma 8.12. *Let X and Y be two sets with finite support to level* 1. *Let a be an atom such that $a\#X, Y$. Then*

$$(X \Rightarrow Y)_{\#a} \cong X_{\#a} \Rightarrow Y_{\#a}.$$

The bijection is given by:

$$f \in (X \Rightarrow Y)_{\#a} \longmapsto f' = \lambda x \in X_{\#a}.f(x),$$
$$g \in X_{\#a} \Rightarrow Y_{\#a} \longmapsto g' = \lambda x \in X.\mathsf{fr}\, b.((b\ a)\cdot g)(x).$$

Proof. There are a number of things to check:

- $f' \in X_{\#a} \Rightarrow Y_{\#a}$. By Theorem 4.7 f' has finite support. Suppose $x \in X_{\#a}$. So $a\#x$ and $a\#f$ then by Theorem 4.7 $a\#f(x)$. The result follows.
- *g' is well-defined and $g' \in X \Rightarrow Y$.* Suppose $x \in X$ and b is fresh (so $b\#x, X, Y, g$). Recall that $a\#X$ and $a\#Y$; by Theorems 4.4 and 2.21 it follows that $(b\ a)\cdot g \in X_{\#b} \Rightarrow Y_{\#b}$. Since $b\#x$ we know $((b\ a)\cdot g)(x)$ is well-defined and $((b\ a)\cdot g)(x) \in Y_{\#b}$. Since $b\#((b\ a)\cdot g)(x)$ by Lemma 6.9 $\mathsf{fr}\, b.((b\ a)\cdot g)(x)$ is well-defined. The result follows.
- $a\#g'$. By Theorem 2.19 it suffices to choose fresh c (so $c\#X, Y, g$) and prove that $(c\ a)\cdot g' = g'$. Choose any $x \in X$. Choose fresh b (so $b\#x, X, Y, g$). By Theorem 6.5 ($\forall$ form)

$$((c\ a)\cdot g')(x) = ((c\ a)\cdot(b\ a)\cdot g)(x).$$

  By Theorem 2.19 $a\#(b\ a)\cdot g$. By Lemma 2.15 and Theorem 2.21 $(c\ a)\cdot(b\ a)\cdot g = (c\ a)\cdot g$. The result follows.

- $f'' = f$. Suppose $f \in (X \Rightarrow Y)_{\#a}$. Unpacking definitions,
$$f'' = \lambda x \in X.\mathsf{fr}\, b.((b\ a)\cdot\lambda x \in X_{\#a}.f(x))(x).$$
Choose any $x \in X$, and choose fresh b (so $b\#x, X, Y, f$). By Theorem 6.5 ($\exists$ form) it suffices to prove
$$((b\ a)\cdot\lambda x \in X_{\#a}.f(x))(x) = f(x).$$
By Theorem 4.4 and 2.21
$$(b\ a)\cdot\lambda x \in X_{\#a}.f(x) = \lambda x \in X_{\#b}.f(x)$$
(recall that $a\#f, X$ and $b\#f, X$). Now $x \in X$ and $b\#x$, thus, $x \in X_{\#b}$. The result follows.
- $g'' = g$. Unpacking definitions,
$$\begin{aligned} g'' &= \lambda x \in X_{\#a}.(\lambda x \in X.\mathsf{fr}\, b.((b\ a)\cdot g)(x))(x) \\ &= \lambda x \in X_{\#a}.\mathsf{fr}\, b.((b\ a)\cdot g)(x). \end{aligned}$$
Choose $x \in X_{\#a}$ and choose fresh b (so $b\#x, X, Y, g$). By Theorem 6.5 ($\exists$ form) it suffices to prove
$$((b\ a)\cdot g)(x) = g(x).$$
We reason as follows:
$$\begin{aligned} ((b\ a)\cdot g)(x) &= (b\ a)\cdot(g((b\ a)\cdot x)) && \text{Theorem 2.33} \\ &= (b\ a)\cdot(g(x)) && a\#x,\ b\#x, \text{ Theorem 2.21} \\ &= g(x) && \text{Theorems 4.7 and 2.21.} \end{aligned}$$
The result follows. ⊣

8.4. Using $X_{\#a}$ to prove properties of $[\mathbb{A}]X$.

LEMMA 8.13. *Let X be a set with finite support to level 1. Let a be an atom such that $a\#X$. Then*
$$[\mathbb{A}](X_{\#a}) = ([\mathbb{A}]X)_{\#a}.$$

PROOF. Suppose $\hat{x} \in [\mathbb{A}](X_{\#a})$. By construction (Definition 8.2) there are two possibilities:

- $\hat{x} = [a]x$ where $x \in X_{\#a}$. Note that $[a]x \in [\mathbb{A}]X$ and by Theorem 3.11 $a\#[a]x$. It follows that $[a]x \in ([\mathbb{A}]X)_{\#a}$.
- $\hat{x} = [a']x$ where $x \in X_{\#a}$ and $a'\#X$. Note that $[a']x \in [\mathbb{A}]X$ and by Theorem 3.11 $a\#[a']x$. It follows that $[a']x \in ([\mathbb{A}]X)_{\#a}$.

Now suppose $\hat{x} \in ([\mathbb{A}]X)_{\#a}$. Again, there are two possibilities:

- $\hat{x} = [a]x$ where $a\#X$ ($a\#[a]x$ is guaranteed by Theorem 3.11). Choose a fresh b (so $b\#x, X$). By part 1 of Lemma 3.12 $\hat{x} = [b](b\ a)\cdot x$. By Theorem 2.19 $a\#(b\ a)\cdot x$. Therefore $[b](b\ a)\cdot x \in [\mathbb{A}](X_{\#a})$.

- $\hat{x} = [a']x$ where $a\#[a']x$ and $a'\#X$. Choose a fresh b (so $b\#x, X$). By part 1 of Lemma 3.12 $\hat{x} = [b](b\ a')\cdot x$. By Theorem 3.11 $a\#x$ so by Theorem 2.19 $a\#(b\ a')\cdot x$. Therefore $[b](b\ a')\cdot x \in [\mathbb{A}](X_{\#a})$. ⊣

THEOREM 8.14. *Let X be a set with finite support to level* 1. *Let a be an atom such that $a\#X$. Then*

$$([\mathbb{A}]X)_{\#a} \cong_f X$$

where f and its inverse, we write it g, are given by:

$$f(\hat{x}) = \hat{x}@a, \qquad g(x) = [a]x.$$

(*By Lemma* 3.17 *$\hat{x}@a$ exists, since we assume $a\#\hat{x}$.*)

PROOF. We must show that the maps map between elements of $([\mathbb{A}]X)_{\#a}$ and elements of X.

- To show that $[a]x \in [\mathbb{A}]X$ we must show that $a\#X$. This is by assumption.
- To show that $\hat{x}@a \in X$ there are two cases:
  - Suppose that $\hat{x} \in [\mathbb{A}]X$ because $\hat{x} = [a]x$ for some $x \in X$ and $a\#X$. By part 2 of Theorem 3.19 $\hat{x}@a = x$ and so $\hat{x}@a \in X$.
  - Suppose that $\hat{x} \in [\mathbb{A}]X$ because $\hat{x} = [a']x$ for some $x \in X$ and $a'\#X$, where a' is different from a. By part 1 of Theorem 3.19 $\hat{x}@a = (a'\ a)\cdot x$. By Lemma 8.4 and Theorem 6.5 ($\forall$ form) since $a\#X$ also $\hat{x}@a \in X$.

We must show that the maps are inverse. This is easy:

- $([a]x)@a = x$ by part 2 of Theorem 3.19.
- $[a](\hat{x}@a) = \hat{x}$ by part 3 of Theorem 3.19. ⊣

One of the observations in [23] is that atoms-abstraction is remarkably well-behaved, and commutes even with function-spaces (see Corollary 9.6.9 of [23] or Remark 5.6 of [37]). We can use a-fresh sets to give some quite slick proofs of these facts:

THEOREM 8.15. *Let X and Y be sets with finite support to level* 1. *Then*

$$\boxed{[\mathbb{A}](X\ op\ Y) \cong [\mathbb{A}]X\ op\ [\mathbb{A}]Y \text{ for } op \text{ one of } \times, +, \cup, \text{ and } \Rightarrow.}$$

In addition, if f is a function-set with finite support such that

$$\bigcup_{i \in dom(f)} supp(i) \text{ is finite} \tag{1}$$

(*the reason for this restriction will become clear in the proof*) *then*

$$\boxed{[\mathbb{A}]\Big(\bigcup_{i \in dom(f)} f(i)\Big) = \bigcup_{i \in dom(f)} [\mathbb{A}]f(i).}$$

PROOF. We give the reasoning for $[\mathbb{A}](X \Rightarrow Y)$ and for indexed unions; the other cases are similar and no harder.

For functions, we reason as follows; we take a fresh (so $a\#X$ and $a\#Y$):

$$\begin{array}{ll} & X \Rightarrow Y \cong X \Rightarrow Y \\ \text{if and only if (Theorem 8.14)} & ([\mathbb{A}](X \Rightarrow Y))_{\#a} \cong ([\mathbb{A}]X)_{\#a} \Rightarrow ([\mathbb{A}]Y)_{\#a} \\ \text{if and only if (Lemma 8.12)} & ([\mathbb{A}](X \Rightarrow Y))_{\#a} \cong ([\mathbb{A}]X \Rightarrow [\mathbb{A}]Y)_{\#a} \\ \text{if and only if (Theorem 8.10)} & [\mathbb{A}](X \Rightarrow Y) \cong [\mathbb{A}]X \Rightarrow [\mathbb{A}]Y. \end{array}$$

Here, the top bijection is the identity, and we take a fresh (so $a\#X, Y$).

Now suppose that f is a function-set with finite support and suppose that $\bigcup_{i \in dom(f)} supp(i)$ is finite. We obtain a bijection between $[\mathbb{A}](\bigcup_{i \in dom(f)} f(i))$ and $\bigcup_{i \in dom(f)} [\mathbb{A}]f(i)$ straight from Lemma 8.11 and Theorem 8.14, but to exhibit an equality we have to do some new calculations.[17] We prove two set inclusions:

- $[\mathbb{A}](\bigcup_{i \in dom(f)} f(i)) \subseteq \bigcup_{i \in dom(f)} [\mathbb{A}]f(i)$.
  Suppose that $[a]x \in [\mathbb{A}](\bigcup_{i \in dom(f)} f(i))$. Then there exists some $i \in dom(f)$ such that $x \in f(x)$ and $a\# \bigcup_{i \in dom(f)} f(i)$. Choose fresh b (so $b\#f, x$ and, because of (1), $b\#i$ for every $i \in dom(f)$). It follows by Theorem 4.4 and Theorem 4.7 that $(b\ a)\cdot x \in \bigcup_{i \in dom(f)} f(i)$. So there exists some $i' \in dom(f)$ such that $(b\ a)\cdot x \in f(i')$. By Theorem 4.7 $b\#f(i')$ so $[b](b\ a)\cdot x \in [\mathbb{A}]f(i')$—(1) helps make sure that $b\#i'$. It follows by part 2 of Lemma 3.12 that $[a]x \in [\mathbb{A}]f(i')$ and the result follows.
- $\bigcup_{i \in dom(f)} [\mathbb{A}]f(i) \subseteq [\mathbb{A}](\bigcup_{i \in dom(f)} f(i))$.
  By Theorem 8.3. ⊣

REMARK 8.16. The sense in which the abstraction type-former commutes with unions given in Theorem 8.15 is slightly more general than that given in [37] (Section 6, page 14, just after equation (50)). Here, we do not insist that f is definable in the language of ZFA. We replace that with the condition that f have finite support and satisfy (1). ◇

REMARK 8.17. We explicitly unpack the bijection in the proof Theorem 8.15 in the cases of $\times$ and $\Rightarrow$:

- $([a]x, [a]y)$ bijects with $[a](x, y)$.
- $\hat{f} \in [\mathbb{A}](X \Rightarrow Y)$ bijects with $\lambda \hat{x} \in [\mathbb{A}]X.\mathsf{fr}\, a.[a](\hat{f}@a)(\hat{x}@a)$
  and
  $f \in [\mathbb{A}]X \Rightarrow [\mathbb{A}]Y$ bijects with $\mathsf{fr}\, a.[a]\lambda x \in X.(f([a]x)@a)$. ◇

Theorems 8.18 and 8.20 characterise the functions out of $[\mathbb{A}]X$ in two different ways.

[17] So we do not use the case of $\bigcup$ in Lemma 8.11, but we retain it for completeness.

THEOREM 8.18. *Let X and Y be sets with finite support to level* 1. *Let a be an atom such that $a\#X$ and $a\#Y$.*
Then

$$([\mathbb{A}]X \Rightarrow Y)_{\#a} \cong X \Rightarrow Y_{\#a}.$$

(Note that by Theorem 8.10 we can settle an equality between X and Y by examining equality between $X_{\#a}$ and $Y_{\#a}$, so in that sense Theorem 8.18 really does characterise all the functions out of $[\mathbb{A}]X \Rightarrow Y$, and not just those that a is fresh for.)

PROOF. By Lemma 8.12 and Theorem 8.14. ⊣

REMARK 8.19. We explicitly unpack the bijection in the proof of Theorem 8.18:

$$f \longmapsto \lambda x \in X.\ f([a]x),$$
$$g \longmapsto \lambda \hat{x} \in [\mathbb{A}]X.\ \mathsf{fr}\, c.(c\ a)\cdot(g(\hat{x}@c)). \qquad \diamond$$

THEOREM 8.20. *Let X be a set with finite support to level* 1, *and let Y be a set with finite support* (*to level* 0). *Define*

$$A = \mathbb{A}\backslash supp(X).$$

Then a bijection

$$\{f \in (\mathbb{A} \times X) \Rightarrow Y \mid \forall x \in X.\mathsf{И}a.a\#f(a,x)\} \cong [\mathbb{A}]X \Rightarrow Y$$

is given by the maps

$$f \in (\mathbb{A} \times X) \Rightarrow Y \longmapsto f' = \lambda \hat{x} \in [\mathbb{A}]X.\mathsf{fr}\, a.f(a, \hat{x}@a),$$
$$g \in [\mathbb{A}]X \Rightarrow Y \longmapsto g' = \lambda(a,x) \in (A \times X).g([a]x).$$

PROOF. There are several things to prove; we consider the less trivial ones:

- *f' is well-defined and is in $[\mathbb{A}]X \Rightarrow Y$.*
  Fix $\hat{x} \in [\mathbb{A}]X$. Choose fresh a (so $a\#f, \hat{x}$). $\hat{x}@a$ exists by Lemma 3.17. $\hat{x}@a \in X$ by Lemma 8.4. So $f(a, \hat{x}@a)$ is well-defined and in Y.
  $\mathsf{И}a.a\#f(a,x)$ is the condition required for $\mathsf{fr}\, a.f(a, \hat{x}@a)$ to be well-defined.
- *$g' \in (A \times X) \Rightarrow Y$ is such that $\forall x \in X.\mathsf{И}a.a\#g'(a,x)$.*
  By Theorem 6.5 ($\exists$ form) it suffices to take any $x \in X$ and some fresh a (so $a\#x, g$) and check that $a\#g([a]x)$. This is by Theorems 3.11 and 4.7.
- *$f'' = f$.*
  By Theorem 6.5 ($\exists$ form) it suffices to check that for any $x \in X$ and some fresh a it is the case that $f(a, ([a]x)@a) = f(a,x)$. This is by part 2 of Theorem 3.19.

- $g'' = g$.
  By Theorem 6.5 ($\exists$ form) it suffices to check that for any $x \in X$ and some fresh a (so $a\#\hat{x}$) it is the case that $g([a](\hat{x}@a)) = g(\hat{x})$. This is by part 3 of Theorem 3.19. $\dashv$

(Theorem 8.20 is slightly more general than a similar result in the literature [37, Lemma 6.3] in the sense that here we consider the more general case where X and Y are not necessarily equivariant.)

§9. Categories arising from $\mathcal{U}$.

$\mathcal{U}$ from Definition 2.4 gives rise to some interesting concrete categories, which we now explore.

The categorical work on names and binding in nominal style has used concrete categories (essentially, the three categories considered below). It remains to develop abstract characterisations in category-theoretic language—that is, using diagrams and universal properties—of the properties of nominal techniques which make it useful. Work in this direction has been carried out first by Menni [50], and recently by Clouston and Pitts [12].

9.1. Three categories: ZFASet, FMSet, and NOMSet.

DEFINITION 9.1. The category ZFASet of *ZFA sets* has objects sets $X, Y \in \mathcal{U}$ (Definition 2.4) and arrows function-sets $f \in X \to Y$ (Definition 2.12) between them.

REMARK 9.2. ZFASet is cartesian closed; the cartesian closed structure is given by $\times$ and $\to$ (Definitions 2.7 and 2.10); a terminal object is a one-element set. It is also a boolean topos; a subobject classifier is $\mathbb{B}$ (Definition 2.8). $\diamond$

We now set about constructing FMSet and NOMSet.

DEFINITION 9.3. Call an element x *equivariant* when $supp(x) = \varnothing$.

For example, a is not equivariant and $\mathbb{A}$ is equivariant.

LEMMA 9.4. *Let X and Y be equivariant sets. Let $f \in X \to Y$ be a function-set. Then*

$$\boxed{\begin{array}{c}f \textit{ is equivariant if and only if } \pi\cdot(f(x)) = f(\pi\cdot x)\\ \textit{for all } x \in X \textit{ and all permutations } \pi.\end{array}}$$

PROOF. If $supp(f) = \varnothing$ the result follows by Theorem 2.33 and Theorem 2.21.

Conversely if $\pi(f(x)) = f(\pi\cdot x)$ for all $x \in X$ and all permutations π then it is easy to verify that $\pi\cdot f = f$ for all permutations π. Thus, $\varnothing$ supports f and the result follows by Theorem 2.21. $\dashv$

DEFINITION 9.5. We obtain two categories from $\mathcal{U}$ in addition to ZFASet:

- The category FMSet of *FM* (*Fraenkel–Mostowski*) *sets*, with objects sets $X \in \mathcal{U}$ with finite support to level 1 (Definition 8.1), and with arrows finitely-supported function-sets between them (elements $f \in X \Rightarrow Y$ from Definition 8.5).
- The category NOMSet of *nominal sets*, with objects equivariant (Definition 9.3) sets $X \in \mathcal{U}$ with finite support to level 1, and with arrows equivariant function-sets between them.

REMARK 9.6. If we write $f\colon X \longrightarrow Y \in \mathsf{FMSet}$ this means: "X and Y are elements of $\mathcal{U}$ with finite support to level 1, and f is a function-set from X to Y with finite support; that is, $f \in X \Rightarrow Y$".

If we write $f\colon X \longrightarrow Y \in \mathsf{NOMSet}$ this means something something a little different: "X and Y are equivariant elements of $\mathcal{U}$ with finite support to level 1, and f is an equivariant function-set from X to Y". $\diamond$

LEMMA 9.7. *Let X, Y, and Z be sets with finite support (to level 0). Then*

$$(X \times Y) \Rightarrow Z \cong_f X \Rightarrow (Y \Rightarrow Z)$$

where f and its inverse, which we write g, are given by

$$f \in (X \times Y) \Rightarrow Z \longmapsto \lambda x \in X.\lambda y \in Y.f(x,y),$$
$$g \in X \Rightarrow (Y \Rightarrow Z) \longmapsto \lambda(x,y) \in X \times Y.(g(x))(y).$$

The bijection restricts to equivariant elements of $(X \times Y) \Rightarrow Z$ and $X \Rightarrow (Y \Rightarrow Z)$.

PROOF. The maps f and g, *currying* and *uncurrying*, are well-known.

We must show that if f has finite support then so does $\lambda x \in X.\lambda y \in Y.f(x,y)$, and similarly for g. This is by Theorem 4.7.

Similarly, if $supp(f) = \varnothing$ then by Theorem 4.7 also $supp(\lambda x \in X.\lambda y \in Y.f(x,y)) = \varnothing$, and similarly for g. $\dashv$

LEMMA 9.8. *If X and Y are equivariant then so is $X \Rightarrow Y$.*

PROOF. By Theorem 4.7. $\dashv$

COROLLARY 9.9. *The categories* FMSet *and* NOMSet *are cartesian closed.*

PROOF. By Lemmas 9.7 and 9.8. Products are described in Definition 2.7. A terminal object is the one-element set $\{\varnothing\}$. $\dashv$

LEMMA 9.10. *Let X be a set with finite support. There is a bijection between finitely-supported subsets U of X, and $X \Rightarrow \mathbb{B}$.*

PROOF. The maps are well-known:

- $f \in X \Rightarrow \mathbb{B}$ corresponds with $U_f = \{x \mid f(x) = 1\} \subseteq X$ and
- $U \subseteq X$ corresponds with $f_U = \{(x,1) \mid x \in U\} \cup \{(x,0) \mid x \in X \setminus U\}$.

(f_U is the graph of $\lambda x \in X.\text{if } x \in U \text{ then } 1 \text{ else } 0$.)

It is easy to show that these maps are inverse, and they map finitely-supported elements with finitely-supported elements by Theorem 4.7. $\dashv$

COROLLARY 9.11. *The categories* FMSet *and* NOMSet *are boolean toposes.*

PROOF. It suffices to show that each of FMSet and NOMSet is cartesian closed and that $\mathbb{B}$ is a subobject classifier.

Cartesian closure is Corollary 9.9. By Lemma 9.10 subobjects of X biject with finitely-supported subsets of X; naturality can also be checked. Finally, $\mathbb{B}$ is clearly a boolean algebra. $\dashv$

9.2. Presheaf presentation of NOMSet the category of nominal sets.

DEFINITION 9.12. Let $\mathcal{I}$ be the category whose objects are finite sets of atoms and whose arrows are injective functions between them.

REMARK 9.13. An important property of support is that if A supports x and B supports x then $A \cap B$ supports x, as described in Theorem 2.21.

This means that if x has a finite supporting set then it has a unique *least* finite supporting set. This key fact allows a compact presentation of a certain collection of presheaves, as sets. We do this in Theorem 9.14.

From the point of view of the presheaf presentation, the important point is that the presheaves should 'preserve sets intersections', or to re-frame this in categorical language, that they preserve *pullbacks of monos*. Every arrow in $\mathcal{I}$ is mono, so this condition is not fully visible in Theorem 9.14.

In [31], we consider presheaves that preserve pullbacks of monos over an indexing category not all of whose arrows are monos, and we use a 'preserves pullbacks of monos' condition to develop a sets-based presentation of the resulting category that is not dissimilar to NOMSet the category of nominal sets. Contrast this with work using presheaves that do not necessarily preserve of pullbacks of monos [19, 18]. $\diamond$

THEOREM 9.14. NOMSet *is equivalent with the category of pullback-preserving functors in* $\mathsf{FMSet}^{\mathcal{I}}$ (*pullback-preserving presheaves on* $\mathcal{I}^{op}$, *to the category of FM sets*).

PROOF. On objects, maps are given by:

- $X \in \mathsf{NOMSet}$ maps to the presheaf $F(X)$ such that
  $$F(X)(A) = \{x \in X \mid supp(x) \subseteq A\} \text{ and } F(X)(f) = \lambda x.\pi{\cdot}x,$$
  where π is any permutation such that $\pi(a) = f(a)$ for all $a \in A$.
- $P \in \mathsf{FMSet}^{\mathcal{I}}$ maps to
  $$G(P) = \{(x, P(A)) \mid x \in P(A)\}/{\sim}$$
  where $\sim$ is the least equivalence relation on pairs $(x, P(A))$ where $x \in P(A)$ such that $(x, P(A)) \sim (y, P(B))$ if there is an injection $\imath\colon A \to B$ such that $y = P(\imath)(x)$.

Consider a $P \in \mathsf{FMSet}^{\mathcal{I}}$ that preserves pullbacks. It follows that P preserves monos and that if $A' \subseteq A$ and $A'' \subseteq A$ then $P(A' \cap A'')$ is isomorphic with $P(A') \cap P(A'')$. Therefore, for every $(x, P(A))$ there is a unique least $C \subseteq A$ and $z \in P(C)$ such that $(x, P(A) \sim (z, P(C))$.

Using this way of choosing representative elements, it is routine to establish equivalences between X and $GF(X)$, and between $FG(P)$ and P. $\dashv$

It can be shown that the category of pullback-preserving presheaves is equivalent with the Schanuel Topos, see [46, Section III.9] or [43, A.21, page 79].

9.3. From NOMSet **to the category of nominal sets, and back.** The notion of a *nominal set* was introduced in [36, Section 3], where it was called 'FM-sets'; it appeared in the journal version of the same paper [37] as '$perm(\mathbb{A})$-sets with the finite support property', and also featured in [58, 23].

DEFINITION 9.15. A *nominal set* $\mathbb{X}$ is a pair $(|\mathbb{X}|, \cdot)$ of a set $|\mathbb{X}| \in \mathcal{U}$ with a finitely-supported $\mathbb{P}$-group action, where:

- A $\mathbb{P}$-group action $\cdot$ is a function
$$\mathbb{P} \times |\mathbb{X}| \to |\mathbb{X}|$$
such that
$$id{\cdot}x = x \quad \text{and} \quad \pi{\cdot}(\pi'{\cdot}x) = (\pi{\circ}\pi'){\cdot}x. \tag{2}$$
- Finite support means there is some finite set of atoms A such that if $\pi \in \mathbb{P}$ and $\pi(a) = a$ for every $a \in A$, then $\pi{\cdot}x = x$, that is, x has finite supporting set in the sense of Definition 2.16.

We call x an *element* of $\mathbb{X}$.

DEFINITION 9.16. Suppose that $\mathbb{X}_1, \dots, \mathbb{X}_n, \mathbb{Y}$ are nominal sets. Call a function
$$f \in (|\mathbb{X}_1| \times \cdots \times |\mathbb{X}_n|) \to |\mathbb{Y}|$$
equivariant when
$$\pi{\cdot}f(x_1, \dots, x_n) = f(\pi{\cdot}x_1, \dots, \pi{\cdot}x_n)$$
for all $x_1 \in |\mathbb{X}_1|, \dots, x_n \in |\mathbb{X}_n|$.

(Definition 9.16 is in harmony with Definition 9.3; see Lemma 9.4.)

DEFINITION 9.17. Let NOM be the category of nominal sets and equivariant functions between them.

We need a subsidiary definition and two lemmas for Theorem 9.21:

DEFINITION 9.18. Suppose $X \in \mathcal{U}$. Define the *transitive closure* $tc(x)$ by:

- $tc(a) = \{a\}$.
- $tc(X) = X \cup \bigcup\{tc(x) \mid x \in X\}$ for $X \notin \mathbb{A}$.

For each set $X \in \mathcal{U}$ make a fixed but arbitrary choice of equivariant element $fresh(X)$ such that $fresh(X) \notin \{X\} \cup tc(X)$. Also choose a fixed but arbitrary bijective map γ from atoms $\mathbb{A}$ to the natural numbers $\mathbb{N}$.

Define an operation $d_u(x)$ (for *deatom*) inductively as follows:

- $d_u(a) = (u, \gamma(a))$ for $a \in \mathbb{A}$.
- $d_u(X) = \{d_u(x) \mid x \in X\}$ for $X \notin \mathbb{A}$.

Intuitively, $d_u(x)$ replaces each atom a with the pair $(u, \gamma(a))$ throughout x.

LEMMA 9.19. *Provided $u \notin tc(X)$, the map $\lambda x \in X.d_u(x)$ bijects X with $d_u(X)$.*

PROOF. By a routine calculation. $\dashv$

LEMMA 9.20. *Suppose u is equivariant. Then so is $d_u(x)$.*

PROOF. By a routine inductive argument using Theorem 2.29. $\dashv$

THEOREM 9.21. NOM *is categorically equivalent with* NOMSet (*Definition* 9.5).

The maps of the equivalence are given by:

- *An equivariant $X \in \mathcal{U}$ in* NOMSet *maps to $F(X) = (X, \bullet)$ where $\pi{\bullet}x = \pi{\cdot}x$.*
  An equivariant function $f \in X \Rightarrow Y \in$ NOMSet *maps to $F(f) = \lambda x.f(x)$.*
- *An object $\mathbb{X} \in$* NOM *maps to $G(\mathbb{X}) = (\mathbb{P} \times d_u(|\mathbb{X}|))/\!\sim$ where $u = fresh(|\mathbb{X}|)$ and*
  $$(\pi, d_u(x)) \sim (\pi', d_u(x')) \textit{ when } \pi{\bullet}x = \pi'{\bullet}x'.$$
  An arrow $g : \mathbb{X} \longrightarrow \mathbb{Y} \in$ NOM *maps to $G(g)$ which maps*
  $$[(\pi, d_u(x))]_\sim \in G(\mathbb{X}) \quad \textit{to} \quad [(\pi, d_v(g(x)))]_\sim$$
  where $u = fresh(|\mathbb{X}|)$ and $v = fresh(|\mathbb{Y}|)$.

PROOF. The proof is easy. We sketch it below. d ensures that atoms in $|\mathbb{X}|$ do not interact 'by accident' with the permutation action when they are viewed as elements in $\mathcal{U}$.[18]

- *$G(g)$ is well-defined*. That is,
  $$(\pi, d_u(x)) \sim (\pi', d_u(x')) \text{ implies } (\pi, d_v(g(x))) \sim (\pi', d_v(g(x'))).$$
  Suppose $\pi{\bullet}x = \pi'{\bullet}x'$. We use equivariance of g to derive
  $$\pi{\bullet}g(x) = g(\pi{\bullet}x) = g(\pi'{\bullet}x') = \pi'{\bullet}g(x').$$
  The result follows.

[18]For example, $\mathbb{A}$ with trivial action $\pi{\bullet}a = a$ always, is an object in NOM. However, the atoms in the underlying set $\mathbb{A}$ are completely irrelevant to the desired permutation action. d removes them to prevent an 'accidental capture' by the permutation action in NOMSet, which would see them and permute them if we allowed it to.

A design alternative would be to insist that $tc(|\mathbb{X}|) \cap \mathbb{A} = \varnothing$, i.e., that the underlying set of a nominal set be a 'normal' set, without atoms—or we could impose the slightly weaker condition of equivariance for all elements of $|\mathbb{X}|$.

However, this would only push d over to F mapping NOMSet to NOM. Our design seems a little more elegant.

- $FG(\mathbb{X}) \cong \mathbb{X}$. We biject $x \in |\mathbb{X}|$ with $[(id, d_u(x))]_\sim$ using Lemma 9.19. We check this bijection is equivariant:

$$\begin{aligned}\pi\cdot[(id, d_u(x))]_\sim &\overset{\text{Thm. 4.4}}{=} [(\pi, \pi\cdot d_u(x))]_\sim \\ &\overset{\text{Lem. 9.20}}{=} [(\pi, d_u(x))]_\sim \\ &\overset{\text{Def.}}{=} [(id, d_u(\pi\bullet x))]_\sim\end{aligned}$$

- $GF(X) \cong X$. We biject $[(\pi, d_u(x))]_\sim$ with $\pi\bullet x$ using Lemma 9.19. $\dashv$

REMARK 9.22. NOM and NOMSet are not quite the same thing, because NOMSet comes equipped with rank induction inherited from $\mathcal{U}$. $\diamond$

9.4. Functors and non-functors on FMSet and NOMSet. $[\mathbb{A}]$- from Definition 8.2 can be viewed as a functor on FMSet. Definition 8.2 gives the action on objects, it remains to specify the action on arrows:

DEFINITION 9.23. If $f\colon X \longrightarrow Y \in \mathsf{FMSet}$ define

$$\boxed{[\mathbb{A}]f = \lambda\hat{x} \in [\mathbb{A}]X.\mathsf{fr}\,a.[a](f(\hat{x}@a)).}$$

$[\mathbb{A}]f$ is well-defined by Theorem 3.11 and Lemma 3.17. By Theorem 4.7 it is finitely-supported, and it is in $[\mathbb{A}]X \Rightarrow [\mathbb{A}]Y$.

REMARK 9.24. Notation is overloaded between Definitions 8.2 and 9.23; between the action on arrows $[\mathbb{A}]f$ in FMSet of Definition 9.23, and the action on sets $[\mathbb{A}]f$ in $\mathcal{U}$ of Definition 8.2.

It will always be clear from the context which is meant. In particular, $[\mathbb{A}]f$ will henceforth always refer to Definition 9.23. $\diamond$

DEFINITION 9.25. If $X \in \mathsf{FMSet}$ define

$$X_{[\mathbb{A}]} = \{h \in \mathbb{A} \Rightarrow X \mid \text{И}a.a\#h(a)\} \in \mathsf{FMSet}.$$

If $f\colon X \Rightarrow Y \in \mathsf{FMSet}$ define

$$f_{[\mathbb{A}]} = \lambda h \in X_{[\mathbb{A}]}.\lambda a \in \mathbb{A}.f(h(a)).$$

By Theorem 4.7 $\text{И}a.a\#f(h(a))$ and it follows that $f_{[\mathbb{A}]}\colon X_{[\mathbb{A}]} \to Y_{[\mathbb{A}]}$. It is not hard to verify that $\text{-}_{[\mathbb{A}]}$ is a functor.

THEOREM 9.26. $\text{-}_{[\mathbb{A}]}$ *is right-adjoint to* $[\mathbb{A}]$- *as an endofunctor on* FMSet.

PROOF. We state the unit, counit, and the correspondence between arrows:

- The unit $\eta_X\colon X \longrightarrow ([\mathbb{A}]X)_{[\mathbb{A}]} \in \mathsf{FMSet}$ is given by

$$\eta_X = \lambda x \in X.\lambda a \in \mathbb{A}.[a]x.$$

- The counit $\epsilon_X\colon [\mathbb{A}](X_{[\mathbb{A}]}) \longrightarrow X$ is given by

$$\epsilon_X = \lambda\hat{f} \in X_{[\mathbb{A}]}.\mathsf{fr}\,a.(f@a)(a).$$

- $f\colon [\mathbb{A}]X \Rightarrow Y$ corresponds with $\lambda x \in X.\lambda a.f([a]x)\colon X \Rightarrow Y_{[\mathbb{A}]}$.
- $g\colon X \Rightarrow Y_{[\mathbb{A}]}$ corresponds with $\lambda \hat{x} \in [\mathbb{A}]X.\mathsf{fr}\, a.g(\hat{x}@a)\colon [\mathbb{A}]X \Rightarrow Y$. $\dashv$

Definition 9.27. Let X and Y be finitely-supported sets to level 1 (Definition 8.1). Define

$$X \otimes Y = \{(x, y) \mid supp\,(x) \cap supp\,(y) = \varnothing\}.$$

$\otimes$ is an endofunctor on NOMSet on both its arguments; we consider only the first:

Definition 9.28. If $f\colon X \longrightarrow X' \in$ NOMSet define

$$f \otimes Y = \lambda(x, y) \in X \otimes Y.(f(x), y).$$

By Theorem 4.7 $supp\,(f(x)) \subseteq supp\,(x)$ so $(f(x), y) \in X' \otimes Y$. It is easy to check that this defines a functorial action from NOMSet to itself.

Remark 9.29. $\text{-} \otimes Y$ is not functorial on FMSet because

$$supp\,(x) \cap supp\,(y) = \varnothing$$

does not imply

$$supp\,(f(x)) \cap supp\,(y) = \varnothing.$$

This is unlike the case for NOMSet, because in FMSet we do not assume that $supp\,(f) = \varnothing$. $\diamond$

Theorem 9.30. *$\text{-} \otimes \mathbb{A}$ is left-adjoint to $[\mathbb{A}]\text{-}$ as an endofunctor on* NOMSet.

Proof. We state the unit, counit, and the correspondence between arrows:

- The unit $\eta_X\colon X \longrightarrow [\mathbb{A}](X \otimes \mathbb{A})$ is given by
$$\eta_X = \lambda x \in X.\mathsf{fr}\, a.[a](x, a).$$
- The counit $\epsilon_X\colon ([\mathbb{A}]X) \otimes \mathbb{A} \longrightarrow X$ is given by
$$\epsilon_X = \lambda(\hat{x}, a) \in ([\mathbb{A}]X) \otimes \mathbb{A}.\hat{x}@a.$$
- $f\colon X \otimes \mathbb{A} \longrightarrow Y$ corresponds with $\lambda x.\mathsf{fr}\, a.[a]f(x, a)$.
- $g\colon X \longrightarrow [\mathbb{A}]Y$ corresponds with $\lambda(x, a).[a]g(x)$. $\dashv$

Remark 9.31. $\text{-}_{\#a}$ from Definition 8.8 is not an endofunctor on FMSet or NOMSet for the following reasons:

- FMSet. Suppose $f\colon X \longrightarrow Y \in$ FMSet and $x \in X$ and $a\#x$. It does not follow that $a\#f(x)$.
- NOMSet. Suppose $X \in$ NOMSet. $a\#X_{\#a}$ is false in general, so $X_{\#a}$ is not in general an object in NOMSet.

$\text{-}_{\#a}$ can be viewed as a functor from NOMSet to FMSet. If $f\colon X \longrightarrow Y \in$ NOMSet define $f_{\#a} = \lambda x \in X_{\#a}.f(x)$. By Theorem 4.7 $a\#f(x)$. More on this in Subsection 9.5. $\diamond$

REMARK 9.32. An easy corollary of Theorem 8.14 is that the full subcategory of FMSet on elements of the form $([\mathbb{A}]X)_{\#a}$, is categorically equivalent to FMSet itself. The relevant maps are given in Theorem 8.14: abstraction $[a]$-: $X \longrightarrow ([\mathbb{A}]X)_{\#a}$, and concretion -$@a$: $([\mathbb{A}]X)_{\#a} \longrightarrow X$. In words: atoms-abstraction is inverse to (fresh) atoms-concretion. $\diamond$

9.5. 'Fresh for' and 'fresh for all elements of' are distinct, but isomorphic. Suppose X is a set in $\mathcal{U}$. Consider the notions 'fresh for' versus 'fresh for all elements of'. In symbols, consider the predicates

$$a\#X \quad \text{versus} \quad \forall x \in X.a\#x.$$

If X is *finite* then 'fresh for' and 'fresh for all elements of' coincide (this is a corollary of Theorem 2.29).

A characteristic feature of nominal techniques is that if X is *infinite* then the two notions part company, and no particular implication connects them in general. For example:

- $a\#\mathbb{A}$ but it is not the case that $a\#x$ for every $x \in \mathbb{A}$ (take $x = a$), and
- it is not the case that $a\#\mathbb{A}\backslash\{a\}$ but it is the case that $a\#x$ for every $x \in \mathbb{A}\backslash\{a\}$.

(This expands on the discussion in Remark 2.17; see also Lemma 8.9.)

So just because $a\#X$ does not mean that a is fresh for every element in X, and conversely, just because $a\#x$ for every $x \in X$ does not mean that a is fresh for X overall.

However, with the mathematics now available to us we can state and prove a different, but still very close, relation between these two notions.

In Definition 9.33 we will define two categories; $\mathsf{FMSet}_{\#\mathsf{a}}$ and $\mathsf{FMSet}_{\nu\mathsf{a}}$. They both implement notions of 'exclude a':

- In $\mathsf{FMSet}_{\#\mathsf{a}}$, a is excluded in the sense of 'fresh for all the elements of'.
- In $\mathsf{FMSet}_{\nu\mathsf{a}}$, a is excluded in the sense of 'fresh for'.

In Theorem 9.42 we will then prove that $\mathsf{FMSet}_{\#\mathsf{a}}$ and $\mathsf{FMSet}_{\nu\mathsf{a}}$ are isomorphic. In this precise sense, being 'fresh for' and being 'fresh for all elements of' are reunited, modulo a natural isomorphism of categories.

DEFINITION 9.33. For the rest of this subsection fix an atom a.

- Define a category $\mathsf{FMSet}_{\#\mathsf{a}}$ by:
  - Objects are elements X of FMSet such that $\forall x \in X.a\#x$.
  - Arrows are arrows $f\colon X \longrightarrow Y \in \mathsf{FMSet}$.
- Define a category $\mathsf{FMSet}_{\nu\mathsf{a}}$ by:
  - Objects are elements X of FMSet such that $a\#X$.
  - Arrows are arrows $f\colon X \longrightarrow Y \in \mathsf{FMSet}$ such that $a\#f$.

DEFINITION 9.34. Suppose X is an object in FMSet. Define $\nu a.X$ by

$$\nu a.X = \{\pi' \cdot x' \mid \pi' \in \mathit{fix}(\mathit{supp}\,(X)\backslash\{a\}),\ x' \in X\}.$$

We read $\nu a.X$ as *restrict a in X* (some reason for this choice of terminology is indicated in Lemma 9.36).

Definition 9.34 is clearly related to Definition 3.1. We can make this formal:

LEMMA 9.35. *Suppose X is an object in* FMSet. *Then*

$$\nu a.X = \bigcup\{x\circlearrowleft_{supp\,(X)\setminus\{a\}} \mid x \in X\}.$$

PROOF. Routine by unpacking definitions (Definitions 3.1 and 9.34). ⊣

LEMMA 9.36. *Suppose X is an object in* FMset.

1. *$supp\,(\nu a.X) \subseteq supp\,(X)\setminus\{a\}$.*
2. *It is not true in general that $supp\,(\nu a.X) = supp\,(X)\setminus\{a\}$.*
3. *$a\#X$ if and only if $\nu a.X = X$.*

PROOF. 1. It is easy to check that if $\pi \in fix(supp\,(X)\setminus\{a\})$ then $\pi\cdot\nu a.X = \nu a.X$. The result follows by Theorem 2.21.

2. It suffices to provide a counterexample. Choose any b and take $X = (\mathbb{A}\times\mathbb{A})\setminus\{(a,b)\}$ (Definition 2.7). It is easy to check that $\nu a.X = \mathbb{A}\times\mathbb{A}$.
3. Suppose $a\#X$. Suppose $\pi' \in fix(supp\,(X)\setminus\{a\})$ and $x' \in X$. By Theorems 4.4 and 2.21, $\pi'\cdot x' \in X$. It follows easily that $\nu a.X = X$.

   Conversely, if $\nu a.X = X$ then by part 1 of this result $a\#X$. ⊣

LEMMA 9.37. 1. *Suppose X is an object in* $\mathsf{FMSet}_{\nu a}$. *Then $\nu a.(X_{\#a}) = X$.*
2. *Suppose X is an object in* $\mathsf{FMSet}_{\#a}$. *Then $(\nu a.X)_{\#a} = X$.*

PROOF. For the first part, we prove two set inclusions:

- *Proof that $X \subseteq \nu a.(X_{\#a})$.* Suppose $x \in X$. Choose some fresh b (so $b\#x, X$). By Theorem 2.21 $(b\ a)\cdot x \in X$ and furthermore by Theorem 2.19 $(b\ a)\cdot x \in X_{\#a}$. It follows from Definition 9.34 that $x \in \nu a.(X_{\#a})$.
- *Proof that $\nu a.(X_{\#a}) \subseteq X$.* Suppose $x \in \nu a.(X_{\#a})$. So $x = \pi'\cdot x'$ for some $\pi' \in fix(supp\,(X_{\#a})\setminus\{a\})$ and some $x' \in X_{\#a}$. By Theorem 4.7 $supp\,(X_{\#a})\setminus\{a\} \subseteq supp\,(X)$. Therefore $\pi' \in fix(supp\,(X))$. Now $x' \in X$, so by Theorem 2.21 $\pi'\cdot x' \in X$.

For the second part, again we prove two set inclusions:

- *Proof that $X \subseteq (\nu a.X)_{\#a}$.* Suppose $x \in X$ (so $a\#x$). Then $x \in \nu a.X$ and it is immediate that $x \in (\nu a.X)_{\#a}$.
- *Proof that $(\nu a.X)_{\#a} \subseteq X$.* Suppose $x \in (\nu a.X)_{\#a}$. So $x = \pi'\cdot x'$ for some $\pi' \in fix(supp\,(X)\setminus\{a\})$ and some $x' \in X$ (so $a\#x'$). Now choose some entirely fresh b (so $b\#X, x, a, \pi'$) and write $\pi = (b\ a)\circ\pi'\circ(b\ a)$. It is a fact that $\pi \in fix(supp\,(X) \cup \{a\})$, so by Theorem 2.21 $\pi\cdot X = X$. Since $a\#x'$, it is also a fact that $\pi|_{supp\,(x')} = \pi'|_{supp\,(x')}$. By Theorem 2.21 $\pi'\cdot x' = \pi\cdot x'$. It follows that $x \in X$, and by Theorem 2.19 it also follows that $a\#x$. ⊣

DEFINITION 9.38. Suppose $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\#\mathsf{a}}$. Define $\nu a.f$ by, for every $x \in \nu a.X$,

$$\boxed{(\nu a.f)(x) = \mathsf{fr}\, b.(b\ a)\cdot f((b\ a)\cdot x).}$$

We read $\nu a.f$ as *restrict a in f*.

LEMMA 9.39. *If $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\#\mathsf{a}}$ then $\nu a.f$ is well-defined and $\nu a.f\colon \nu a.X \longrightarrow \nu a.Y \in \mathsf{FMSet}_{\nu\mathsf{a}}$.*

Furthermore, $supp\,(\nu a.f) \subseteq supp\,(f)\setminus\{a\}$.

PROOF. First, we review what has to be proved. Suppose $x \in \nu a.X$. Choose some fresh b (so $b\#f, a, x$). By Theorem 6.5 ($\exists$ form), to calculate $\nu a.f$ it suffices to calculate $(b\ a)\cdot f((b\ a)\cdot x)$, and this must be well-defined. If so, then Definition 6.8 requires us to check that $b\#(b\ a)\cdot f((b\ a)\cdot x)$. We must also check that $supp\,(\nu a.f)$ is finite, and $a\#\nu a.f$; there is no need for a separate proof of this, since it is subsumed by a proof that $supp\,(\nu a.f) \subseteq supp\,(f)\setminus\{a\}$.

We sketch each part of the proof in turn:

- $(b\ a)\cdot f((b\ a)\cdot x)$ *well-defined.* It follows by Theorem 2.19 that $a\#(b\ a)\cdot x$.

  What is slightly non-trivial here is to prove that if $x \in \nu a.X$ then $(b\ a)\cdot x \in X$. Since $x \in \nu a.X$ and $X \in \mathit{FMSet}_{\#a}$, there exists some $\pi' \in \mathit{fix}(supp\,(X)\setminus\{a\})$ and some $x' \in X$ such that $x = \pi'\cdot x'$ and $a\#x'$. We reason as follows:

  $$\begin{aligned} (b\ a)\cdot x &\overset{\text{fact}}{=} (b\ a)\cdot\pi'\cdot x' \\ &\overset{\text{Lem. 2.15}}{=} ((b\ a)\circ\pi'\circ(b\ a))\cdot(b\ a)\cdot x' \\ &\overset{a,b\#x',\text{Thm. 2.21}}{=} ((b\ a)\circ\pi'\circ(b\ a))\cdot x'. \end{aligned}$$

  It is a fact that $((b\ a)\circ\pi'\circ(b\ a)) \in \mathit{fix}(supp\,(X))$ and it follows by Theorems 4.4 and 2.21 that $(b\ a)\cdot x \in X$.
- $b\#(b\ a)\cdot f((b\ a)\cdot x)$. Using Theorem 2.19.
- $b\#(b\ a)\cdot f((b\ a)\cdot x) \in \nu a.Y$. It is a fact that $f((b\ a)\cdot x) \in Y$. The result follows by the definition of $\nu a.Y$.

We now prove that $supp\,(\nu a.f) \subseteq supp\,(f)\setminus\{a\}$. By Theorem 4.7 $supp\,(\nu a.f) \subseteq supp\,(f) \cup \{a\}$. By Theorems 6.2 and 6.5 ($\exists$ form) it then suffices to check that $(a'\ a)\cdot(\nu a.f) = \nu a.f$ for some fresh a' (so $a'\#f, a$). Choose some $x \in \nu a.X$ and some fresh b. We reason as follows:

$$\begin{aligned} &((a'\ a)\cdot\nu a.f)(x) \\ &\quad = (b\ a)\cdot((a'\ a)\cdot f)((b\ a)\cdot(a'\ a)\cdot x) && \text{Theorem 6.5 } (\forall \text{ form}) \\ &\quad = (b\ a)\cdot(a'\ a)\cdot f((b\ a)\cdot x) && \text{Theorem 2.33 and Lemma 2.15} \\ &\quad = (b\ a)\cdot f((b\ a)\cdot x). && \text{Theorems 4.7 and 2.19} \\ &\quad = (\nu a.f)(x) && \text{Theorem 6.5 } (\exists \text{ form}) \quad \dashv \end{aligned}$$

Recall from Definition 8.8 that if $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\nu\mathsf{a}}$ then $f_{\#a}$ is defined by $f_{\#a} = \lambda x \in X_{\#a}.f(x)$.

LEMMA 9.40. *If* $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\nu\mathsf{a}}$ *then* $f_{\#a}\colon X_{\#a} \longrightarrow Y_{\#a} \in \mathsf{FMSet}_{\#\mathsf{a}}$.

PROOF. Suppose $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\nu\mathsf{a}}$. In particular, $a\#f$. It follows using Theorem 4.7 that $f_{\#a}\colon X_{\#a} \longrightarrow Y_{\#a}$. ⊣

LEMMA 9.41. *The following data specifies a pair of functors between* $\mathsf{FMSet}_{\nu\mathsf{a}}$ *and* $\mathsf{FMSet}_{\nu\mathsf{a}}$:

- $\text{-}_{\#a}\colon \mathsf{FMSet}_{\nu\mathsf{a}} \longrightarrow \mathsf{FMSet}_{\#\mathsf{a}}$ *maps* X *to* $X_{\#a}$ *and* $f\colon X \longrightarrow Y$ *to* $f_{\#a}\colon X_{\#a} \to Y_{\#a}$.
- $\text{-}_{\nu a}\colon \mathsf{FMSet}_{\#\mathsf{a}} \longrightarrow \mathsf{FMSet}_{\nu\mathsf{a}}$ *maps* X *to* $\nu a.X$ *and* $f\colon X \longrightarrow Y$ *to* $\nu a.f\colon \nu a.X \to \nu a.Y$.

PROOF. By routine calculations. ⊣

THEOREM 9.42. $\text{-}_{\#a}$ *and* $\text{-}_{\nu a}$ *define an isomorphism of categories between* $\mathsf{FMSet}_{\#\mathsf{a}}$ *and* $\mathsf{FMSet}_{\nu\mathsf{a}}$.

PROOF. That the functors are inverse on objects follows quickly from Lemma 9.37.

We check that these functors are inverse on arrows. Suppose $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\nu\mathsf{a}}$. So $f \in X \to Y$ and $a\#f, X, Y$. We must check that $\nu a.(f_{\#a}) = f$. Take any $x \in X$ and choose some entirely fresh b. We reason as follows:

$$\begin{aligned}(\nu a.(f_{\#a}))(x) &= (b\ a)\cdot f((b\ a)\cdot x) && \text{Definition 9.34, Theorem 6.5 ($\forall$ form)}\\ &= f((b\ a)\cdot(b\ a)\cdot x) && \text{Theorems 4.4 and 2.21}\\ &= f(x) && \text{Lemma 2.15.}\end{aligned}$$

Suppose $f\colon X \longrightarrow Y \in \mathsf{FMSet}_{\#\mathsf{a}}$. So $f \in X_{\#a} \to Y_{\#a}$. We must check that $(\nu a.f)_{\#a} = f$. Take any $x \in X_{\#a}$ and choose some entirely fresh b. We reason as follows:

$$\begin{aligned}(\nu a.f)_{\#a}(x) &= (b\ a)\cdot f((b\ a)\cdot x) && \text{Definition 9.34}\\ &= (b\ a)\cdot f(x) && \text{Theorem 2.21}\\ &= f(x) && \text{Theorems 4.7 and 2.21.}\end{aligned}$$ ⊣

§10. Two set theories: ZFA and FM. Having considered the categories relevant to nominal techniques in Section 9, we now consider the relevant axiomatic set theories. These are Fraenkel–Mostowski set theory (*FM*) and Zermelo–Fraenkel set theory with atoms (*ZFA*). FM and ZFA are logical theories (sets of axioms) in first-order logic, in a signature with

- one constant symbol $\mathbb{A}$ and
- two binary atomic predicate symbols $=$ (equality) and $\in$ (set membership).

Fraenkel–Mostowski set theory was originally created to prove the independence of the axiom of choice from the other axioms of set theory [6]. It was rediscovered and applied by the author and Pitts to the problem of syntax-with-binding [23, 37].

Unfortunately, in [37, Section 1] we gave the impression that a change of foundations is necessary to carry out nominal techniques. This is not so. It is useful to work in ZFA rather than Zermelo–Fraenkel set theory (without atoms), because this gives us equivariance. It is not necessary to insist, as FM does, that *all* elements have finite support. Thus, the body of this paper has been conducted in the cumulative hierarchy model $\mathcal{U}$, and this admits elements without finite support (Remark 2.18).

10.1. Axioms of the two theories.

DEFINITION 10.1. *Terms* t and *formulas* P, Q are defined by:

$$t ::= x, y, z, \cdots \mid \mathbb{A}$$
$$P, Q ::= t \in t \mid t = t \mid P \Rightarrow P \mid \bot \mid \forall x.P$$

$\forall$ is the only binder; we identify formulae up to α-equivalence as is standard.

We will write:

$\neg P$ for $P \Rightarrow \bot$		$P \wedge Q$ for $\neg(P \Rightarrow \neg Q)$	
$P \vee Q$ for $(\neg P) \Rightarrow Q$		$P \Leftrightarrow Q$ for $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$	
$\top$ for $\bot \Rightarrow \bot$		$\exists x.P$ for $\neg\forall x.\neg P$	

Symbols bind in the following precedence, with the leftmost binding tightest:

$\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \forall, \exists.$

$\Rightarrow$ associates to the right. The scope of $\forall$ and $\exists$ extends as far to the right as possible.

DEFINITION 10.2. A *theory* is a set of formulas, which we call *axioms*.

DEFINITION 10.3. The axioms of Fraenkel–Mostowski set theory are listed in Figure 2. The axioms of Zermelo–Fraenkel set theory are as for Fraenkel–Mostowski set theory but without (**AtmInf**) and without (**Fresh**).

REMARK 10.4. In Figure 2 we use the following shorthand:

$\forall x, y.\phi$	is short for $\forall x.\forall y.\phi$,
$\forall x \in y.\phi$	is short for $\forall x.\ x \in y \Rightarrow \phi$,
$x = \{z \mid z \in x\}$	is short for $\forall y.(\forall z.z \in x \Leftrightarrow z \in y) \Rightarrow x = y$,
$y = \{z \in x \mid \phi(z)\}$	is short for $\forall z.z \in y \Leftrightarrow (z \in x \wedge \phi(z))$,
$z = \{\chi(y) \mid y \in x\}$	is short for $\forall u.u \in z \Leftrightarrow \exists y.(\chi(y) = u \wedge y \in x)$,
$z = \{x, y\}$	is short for $\forall u.u \in z \Leftrightarrow (u = x \vee u = y)$,
$z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\}$	is short for $\forall y.y \in z \Leftrightarrow \exists y'.(y \in y' \wedge y' \in x)$,
$z = \{y \mid y \subseteq x\}$	is short for $\forall y.y \in z \Leftrightarrow \forall y'.(y' \in y \Rightarrow y' \in x)$,
$\varnothing \in x$	is short for $\exists z.z \in x \wedge \forall z'.z' \notin z$,
$y \cup \{z\} \in x$	is short for $\exists u.u \in x \wedge \forall u'.(u' \in u \Leftrightarrow u \in y \vee u = z)$.

(Sets)	$\forall x, y.\ y \in x \Rightarrow x \notin \mathbb{A}$	
(Extensionality)	$\forall x.\ x \notin \mathbb{A} \Rightarrow x = \{z \mid z \in x\}$	
(Comprehension)	$\forall x.\exists y.\ y \notin \mathbb{A} \wedge y = \{z \in x \mid \phi(z)\}$	(y not free in ϕ)
($\in$-Induction)	$\big(\forall x.(\forall y \in x.\phi(y)) \Rightarrow \phi(x)\big) \Rightarrow \forall x.\phi(x)$	
(Replacement)	$\forall x.\exists z.\ z \notin \mathbb{A} \wedge z = \{\chi(y) \mid y \in x\}$	
(Pairset)	$\forall x, y.\exists z.\ z = \{x, y\}$	
(Union)	$\forall x.\exists z.\ z \notin \mathbb{A} \wedge z = \{y \mid \exists y'.(y \in y' \wedge y' \in x)\}$	
(Powerset)	$\forall x.\exists z.\ z = \{y \mid y \subseteq x\}$	
(Infinity)	$\exists x.\ \varnothing \in x \wedge \forall y \in x.\ y \cup \{y\} \in x$	
(AtmInf)	$\exists z.\ \mathbb{A} \notin z \wedge (\forall a \in \mathbb{A}.\forall x \in z.\ x \cup \{a\} \in z)$	
(Fresh)	$\forall x.\mathsf{И}b.\mathsf{И}a.\ ((b\ a)\cdot x = x)$	

FIGURE 2. Axioms of FM.

The permutation action $\pi\cdot x$ is as in Definition 2.14. The notion of support is as in Definition 2.16. The $\mathsf{И}$-quantifier is as in Definition 6.1. Permutation, support, and $\mathsf{И}$ can all be formalised in the language of FM. The χ in the axiom **(Replacement)** denotes a function definable in the language of FM, possibly with parameters, in the usual way. $\diamond$

REMARK 10.5. Our intended model for ZFA is the cumulative powerset hierarchy built up from $\mathbb{A}$ the collection of atoms. This is $\mathcal{U}$ from Definition 2.4. The cumulative powerset of finitely-supported sets built up from $\mathbb{A}$ is the intended model of FM. We make this formal in Section 10.2. $\diamond$

10.2. Relative consistency of FM with respect to ZFA. Recall from Definition 10.3 the definitions of ZFA and FM. By Gödel's incompleteness we cannot prove 'FM is consistent'. We can prove a relative consistency result: if ZFA has a model, then so does FM.

Consistency of ZFA relative to Zermelo–Fraenkel set theory is well-known; see for example [40, Problem 1, Chapter 4].

We constructed a model of ZFA in Definition 2.4.[19]

DEFINITION 10.6. Let the *hereditarily finitely supported sets* $\mathcal{HFS}$ be inductively defined by:

> If U is finitely supported and $\forall u \in U.u \in \mathcal{HFS}$, then U is hereditarily finitely supported.

For example $\varnothing \in \mathcal{HFS}$ and if $a \in \mathbb{A}$ then $a \in \mathcal{HFS}$ (because a and $\varnothing$ are finitely supported, and so are all their elements, of which they have none). Also, $\mathbb{A}$ (the set of all atoms) is in $\mathcal{HFS}$. The set *comb* (Remark 2.18) is not in $\mathcal{HFS}$ because it is not finitely supported.

[19] If ZFA is inconsistent then $\mathcal{U}$ does not exist. Thus, this is a *relative* consistency result. Yet if $\mathcal{U}$ does not exist, then our ontological commitment to sets and powersets is wrong. In that case, how can we and the reader be doing mathematics at all? Such questions are outside the scope of our current enquiry.

Definition 10.7. Define pow_{fs} (*fs* for *finitely supported*) by:

$$pow_{fs}(x) = \{y \mid y \subseteq x,\ y \text{ is finitely supported}\}.$$

Lemma 10.8. *Suppose that* $x \in \mathcal{HFS}$. *Then*:

1. $pow_{fs}(x) = \{y \subseteq x \mid y \in \mathcal{HFS}\}$.
2. $pow_{fs}(x) \in \mathcal{HFS}$.

Proof. Suppose that $x \in \mathcal{HFS}$.

1. Suppose $y \in pow_{fs}(x)$. By construction y has finite support. Also, if $y' \in y$ then $y' \in x$ and so $y' \in \mathcal{HFS}$. Therefore $y \in \mathcal{HFS}$.
   Conversely, if $y \in \mathcal{HFS}$ then y has finite support. The equality follows.
2. $x \in \mathcal{HFS}$ so x has finite support. By Theorem 4.7 also $pow_{fs}(x)$ has finite support. We have just shown that all $y \in pow_{fs}(x)$ are in $\mathcal{HFS}$. It follows that $pow_{fs}(x) \in \mathcal{HFS}$. ⊣

Lemma 10.9. $\mathcal{HFS}$ *is a model of FM*.

Proof. We observed that $\mathbb{A} \in \mathcal{HFS}$. We interpret the term-language of FM in the class $\mathcal{HFS}$ in the natural way. It remains to show that $\mathcal{HFS}$ validates the axioms in Figure 2. This is by a routine verification using Theorem 4.7 and Lemma 10.8. For more details see [23, Subsection 11.5] for the full proof, or [40, Chapter 4] for a more general result of which this lemma is an instance. ⊣

Theorem 10.10. *FM is consistent with respect to ZFA*.

Proof. If ZFA is consistent then build $\mathcal{U}$ (Definition 2.4) as discussed at the start of this subsection. By Lemma 10.9 we obtain a model of FM. ⊣

10.3. Fraenkel–Mostowski set theory and choice.

10.3.1. *The axiom of choice.* Given an element z write $\bigcup z = \{z'' \mid \exists z' \in z.\ z'' \in z'\}$. The *axiom of choice* is

$$\textbf{(AC)} \qquad \forall z.(z \neq \varnothing \wedge z \cap (\mathbb{A} \cup \varnothing) = \varnothing) \Rightarrow \exists f \in (z \to \bigcup z).\forall x \in z.f(x) \in x.$$

Intuitively, f is a function-set (Definition 2.10) that picks out an element from x, for every $x \in z$.

The original application of FM was to prove the independence of **(AC)** from the other axioms of ZFA [21]. Four decades later Cohen proved AC was independent from set theory without atoms (ZF), via his forcing method [13]. Also see [40] for a survey.

Theorem 10.11. **(AC)** *is inconsistent with FM*.

Proof. Let X be the set of sets of unordered pairs of distinct atoms. Suppose there exists a choice function f for X.

By **(Fresh)** the choice function f has finite support. Choose $a, b\#f$. By Theorems 4.4 and 2.21, $f(\{b,a\})$ is fixed by $(b\ a)$. However, neither

$b \in \{b, a\}$ nor $a \in \{b, a\}$ is fixed by $(b\ a)$, so f cannot be a choice function; a contradiction.

Therefore f does not exist. $\dashv$

REMARK 10.12. In spite of Theorem 10.11, nominal techniques are consistent with the axiom of choice.

In Definition 2.4 we constructed $\mathcal{U}$, the standard cumulative hierarchy model of ZFA. ZFA is consistent with the axiom of choice. We insist on finite support when we need it; what Theorem 10.11 uses is the fact that a choice function for a finitely-supported set, need not itself be finitely-supported.

There is no overall inconsistency here—just, as is so often the case, some constructions which make sense when applied to elements with the right properties, and may not make sense otherwise. (One cannot reasonably argue, for example, that addition is inconsistent with the existence of elements that are not numbers.)

FM may be convenient in situations where everything we wish to consider will have finite support. (Just as, for example, arithmetic is convenient in situations where we happen to know that we wish to consider numbers.) $\diamond$

10.3.2. *Hilbert's choice.* Careful formulations of the definition of capture-avoiding substitution may use a choice function for picking out fresh variables; an example is [62, Section 2]. If we work in Fraenkel–Mostowski set theory then we should not do this; see Theorem 10.13. As we have seen, the use of atoms as a primitive concept, and the И-quantifier and 'fresh' are well-suited to replace ϵ for this task, and indeed they bring benefits exemplified by equivariance (Theorem 4.4) support (Theorems 2.21 and 4.7) and the some/any property (Theorem 6.5).

We can still use ϵ when we want the full power of choice, but this takes us into Zermelo–Fraenkel set theory with atoms.

THEOREM 10.13. *It is inconsistent to extend the language and axioms of FM with* Hilbert's choice.

That is, we extend the term-language with a binder $\epsilon x.\phi(\overline{z}, x)$, *extend existing axiom-schemes to include the resulting new terms and predicates, and add a new schema of axioms*

$$(\exists x.\phi(\overline{z}, x)) \Rightarrow \phi(\overline{z}, \epsilon x.\phi(\overline{z}.x)).$$

PROOF. By an easy adaptation of the proof of Theorem 10.11. $\dashv$

10.3.3. *Unique choice.* Unique choice is not a problem:

DEFINITION 10.14. Suppose that $\phi(\overline{z}, x)$ is a formula on x and a list of variables $\overline{z}$. As is standard, write $\exists! x.\phi(\overline{z}, x)$ for

$$(\exists x.\phi(\overline{z}, x)) \wedge \forall x, x'.(\phi(\overline{z}, x) \wedge \phi(\overline{z}, x') \Rightarrow x = x').$$

Read this as 'there is a *unique* x such that $\phi(\overline{z}, x)$'.

THEOREM 10.15. *It is consistent to extend the language and axioms of FM with* unique choice (*or* definite description).

That is, we extend the term-language with a binder $\iota x.\phi(\overline{z}, x)$, extend existing axiom-schemes to include the resulting new terms and predicates, and add a new schema of axioms

$$(\exists! x.\phi(\overline{z}, x)) \Rightarrow \phi(\overline{z}, \iota x.\phi(\overline{z}, x)).$$

SKETCH PROOF. Take some model of FM. Suppose $\phi(\overline{z}, x)$ is a formula and suppose some values for $\overline{z}$. There are two cases:

- If $\exists! x.\phi(\overline{z}, x)$ is false then interpret $\iota x.\phi(\overline{z}, x)$ by $\varnothing$.
- If $\exists! x.\phi(\overline{z}, x)$ is true then interpret $\iota x.\phi(\overline{z}, x)$ as the unique element making $\phi(\overline{z}, x)$ true.

With this interpretation, $\exists! x.\phi(\overline{z}, x) \Rightarrow \phi(\overline{z}, \iota x.\phi(\overline{z}, x))$ is always true. $\dashv$

§11. Extensions of the semantics. In this paper we have marked out a certain body of work, spanning eight years of research. This is, however, not the whole story. In this section we indicate, in broad strokes, answers to three questions which are often asked of nominal techniques:

1. What about infinitary syntax, and what about atoms-abstraction over several (or infinitely many) atoms at a time?
2. Can we base a theory of names on renamings (not necessarily bijective maps on atoms), rather than on permutations, and if so, what does it look like?
3. What about substitution?

11.1. Infinitely many names. We investigate the $\reflectbox{N}$ quantifier and the model of name-abstraction in Subsections 6.1 and 3.2 on a general assumption of finite support. However, infinitary syntax may generate elements that do not have finite support; for example an infinite tuple of distinct atoms $(a, b, c, \dots)$ does not have finite support.

The central question is: what sense of 'small' should replace the word 'finite' in 'finite supporting set of atoms' used in this paper?

One obvious answer is to replace 'finite' with 'countable' (and to take the set of all atoms to be uncountable).

Another answer is to replace 'finite' with a notion of *support ideal* introduced by Cheney [10]. A support ideal is a set of sets of atoms $\mathcal{I}$ that does not contain $\mathbb{A}$, is closed under sets unions, is down-closed with respect to subset inclusion, and contains all the finite sets of atoms. That is, a support ideal $\mathcal{I}$ is such that every $S \in \mathcal{I}$ is a subset $S \subseteq \mathbb{A}$ and furthermore:

- $\mathbb{A} \notin \mathcal{I}$.
- If $S \in \mathcal{I}$ and $T \in \mathcal{I}$ then $S \cup T \in \mathcal{I}$.
- If $S \subseteq T$ and $T \in \mathcal{I}$ then $S \in \mathcal{I}$.
- If S is finite then $S \in \mathcal{I}$.

Support ideals distil in axiomatic form those properties of finite sets of atoms that are actually needed when we construct atoms-abstraction and nominal abstract syntax; a support ideal has enough of the behaviour of the set of finite sets of atoms that 'nominal' constructions can still be carried out.

Another answer, developed in [24] and then in [26], is to replace 'finite' with 'well-orderable'. That is, we identify the essential and useful quality of a 'finite' set of atoms, with being well-orderable. In particular, this allow us also to generalise atoms-abstraction and permutation to infinitely many atoms. We use well-orderings to build permutations to rename well-orderable sets of atoms (we might do this because, for example, we are proving a generalised version of Theorem 2.21 and we want to rename infinitely many atoms to be fresh). See [26] for details.

11.2. Permutations vs. renamings and function-spaces. Recall from Definition 9.12 that we write $\mathcal{I}$ for the category of finite sets of atoms and injections between them. Write Set for the category of all sets and functions between them. Write also $\mathcal{F}$ for the category of finite sets and all (not necessarily injective) functions between them.

Higher-order abstract syntax (*HOAS*) [55] uses functions to represent syntactic abstraction using types, in particular the so-called *weak HOAS* (as exemplified for example in [15]) uses types like

$$lam\colon (\mathbb{A} \to tm) \to tm.$$

These require semantic or proof-theoretic justifications.

In this paper we have discussed the category NOM of nominal sets. From Theorem 9.14 and Theorem 9.21 it follows that this can be presented as the category of pullback-preserving presheaves in $\mathsf{FMSet}^{\mathcal{I}}$.[20] Fiore et al. [19] and Hofmann [38] proposed to use $\mathsf{Set}^{\mathcal{F}}$ as a semantic foundation for higher-order abstract syntax. Both can be used as mathematical models for inductive specification and reasoning on syntax with binding.

In $\mathsf{Set}^{\mathcal{F}}$ for a presheaf F, the presheaf F^+ given by $F^+(X) = F(X \cup \{x\})$ for $x \notin X$, is isomorphic with the exponential $\mathbb{A} \Rightarrow F$. In the category NOM of nominal sets the corresponding presheaf is isomorphic with our construction $[\mathbb{A}]X$, with $[\mathbb{A}]$- being the right adjoint to a tensor product different from cartesian product (see Theorem 9.30). In either case $\mathbb{A}$ is given as a presheaf by $\mathbb{A}(S) = S$. Thus, $\mathsf{Set}^{\mathcal{F}}$ seems better-suited for modelling higher-order abstract syntax and its typings $lam\colon (\mathbb{A} \to tm) \to tm$ for variable binding constructs (in this case: 'lambda'). This arises when using an existing theorem prover or type theory (for example Coq) to model higher-order abstract syntax [15, 39].

[20] We write FMSet and not Set so as to account for atoms. In fact we could take Set to be equal to FMSet, either by foundationally admitting that 'atoms actually exist', or alternatively by declaring 'secretly, atoms were sets, for example, they were numbers all along'. The distinction between FMSet and Set is more a matter of taste than of mathematical substance but we maintain it, for consistency.

In [31] we propose a sets-based presentation of the full subcategory of $\mathsf{Set}^{\mathcal{F}}$ of functors preserving pullbacks of monos. We present it as a category of sets with a finitely-supported not-necessarily-injective functions on atoms (we call it a *renaming action*). This is to be contrasted with the work in this paper, which is based on finitely-supported injective functions on atoms (*permutations*).

The pullback requirement implies that $F(X \cap Y)$ is isomorphic with 'the intersection of' $F(X)$ and $F(Y)$, in line with the intuition 'objects with free variables from X' for $F(X)$. This rules out artefacts present in $\mathsf{Set}^{\mathcal{F}}$ like $F(X) = \mathit{if}\ |X| > 1\ \mathit{then}\ \{\star\}\ \mathit{else}\ \varnothing$, which never arise as denotations of types in higher-order abstract syntax.

We also characterise the function-space of nominal renaming sets as a set of functions, rather than a Kripke exponential transported along the equivalence, and we identify the tripos structure needed to reason about renaming sets in a robust way.

(Staton and Fiore [20] define a category of substitution sets equivalent with a sheaf subcategory of $\mathsf{Set}^{\mathcal{F}}$. Their category is not defined concretely in terms of ordinary sets and it does not make $\mathsf{Set}^{\mathcal{F}}$ easier to work with.)

11.3. Substitution on the sets hierarchy. Given an inductive datatype we may define a substitution action on it. Of what general mathematical entity are these substitution actions instances?

Two related answers are given in [33, 19]. These study abstract notions of *substitution algebra* as an algebraic structure (parameterised over a signature) independently of an inductively-defined carrier set; [33] uses *nominal algebra* to do this (a purpose-built logic for expressing axioms with binding introduced in Mathijssen's thesis [48], with a semantics in nominal sets), whereas [19] expresses its axiomatisation as a diagram-scheme, parameterised over binding signatures, of diagrams in $\mathsf{Set}^{\mathcal{F}}$.

In [30] we take a direct approach which exists at the level of the cumulative hierarchy itself. We define a substitution action directly on the powerset constructor, that is, we show how given any finitely-supported set Z, atom a, and element x, we can define a set $Z[a \mapsto x]$ which means in some sensible way 'Z with a replaced by x'.

We now briefly sketch why this is non-trivial in terms of our cumulative hierarchy $\mathcal{U}$.

We want our new substitution to coincide with 'usual' substitution on elements representing syntax (i.e., representing labelled trees). Atoms model variable symbols, therefore we let

$$a[a \mapsto x] = x \text{ and } b[a \mapsto x] = b.$$

Recall from Subsections 3.1 and 3.2 that the atoms-abstraction is formed by taking an α-equivalence class of 'renamed variants' of a set. For example the abstraction of the atom a in the ordered pair (a, b), written $[a](a, b)$

(Definition 3.8) is

$$[a](a,b) = \{(a,(a,b)),\ (c,(c,b)),\ (d,(d,b)),\ (e,(e,b)),\ (f,(f,b)),\dots\}.$$

Also consider the α-abstraction of a in the ordered 3-tuple (a,b,d): $[a](a,b,d)$ (here (z_1,z_2,z_3) is $(z_1,(z_2,z_3))$):

$$[a](a,b,d) = \{(a,(a,b,d)),\ (c,(c,b,d)),\ (e,(e,b,d)),\ (f,(f,b,d)),\dots\}.$$

Note that

$$(d,(d,b)) \in [a](a,b) \text{ and } (d,(d,b,d)) \notin [a](a,b,d)$$

because d does not 'clash' with (a,b) but it does 'clash' with (a,b,d).

We want substitution to distribute over abstraction in a capture-avoiding way. That is, we want

$$([a](a,b))[b \mapsto (b,d)] = [a](a,b,d).$$

We cannot obtain this by operating *pointwise* on the elements of $[a](a,b)$, i.e., by letting

$$Z[a \mapsto x] = \{z[a \mapsto x] \mid z \in Z\}$$

always, for in that case, we would not know *not* to put into

$$([a](a,b))[b \mapsto (b,d)]$$

the element

$$(d,(d,b))[b \mapsto (b,d)] = (d,(d,b,d)).$$

Substitution must detect α-equivalence classes and adjust for them by adding or removing elements.

This is the major difficulty of defining substitution on powersets.

Not all sets look like the tame examples we selected above. Elements may be (amongst many other things) unions of abstractions

$$[a](a,b) \cup [a](a,b,d)$$

or equivalence classes which represent simultaneous abstraction by more than one atom

$$\begin{aligned}\big\{\ &\{a,b,c\},\ \{d,b,c\},\ \{e,b,c\},\ \{f,b,c\},\dots\\ &\{a,d,c\},\ \{b,d,c\},\ \{e,d,c\},\ \{f,d,c\},\dots\\ &\{a,e,c\},\ \{b,e,c\},\ \{d,e,c\},\ \{f,e,c\},\dots\big\}\end{aligned}$$

—we can think of this as 'abstract a and b simultaneously (in no particular order) in $\{a,b,c\}$'.

If we apply $[c \mapsto (b,d)]$ to this we want to 'avoid name-clash' with b and d, and 'α-convert b', and 'fill in renamed variants mentioning c', to obtain the element

$$\begin{aligned}\big\{\ &\{a,c,(b,d)\},\ \{d,c,(b,d)\},\ \{e,c,(b,d))\},\ \{f,c,(b,d)\},\ \dots\\ &\{a,e,(b,d)\},\ \{c,e,(b,d)\},\ \{f,e,(b,d)\},\ \{g,e,(b,d)\},\ \dots\\ &\{a,e,(b,d)\},\ \{c,e,(b,d)\},\ \{f,e,(b,d)\},\ \{g,e,(b,d)\},\ \dots\big\}\end{aligned}$$

which we can think of as 'abstract a and b simultaneously in $\{a,b,(b,d)\}$'.

We note the same issue with interaction with equivalence classes; $\{a,b,(b,d)\}$ is not in this set, so substitution cannot be defined in a purely pointwise manner.

Our strategy for defining substitution as an action purely on sets of elements is as follows:

> Given a set X, identify X as a union of some generalised notion of α-equivalence classes $X = \bigcup_i X_i$. Now carry out substitution on the individual classes, then put the results together again using sets union.

We can put it another way:

> Substitution acts pointwise on (generalised) α-equivalence classes of elements.

Our generalisation of α-equivalence classes is $u\circlearrowleft_A$ from Subsection 3.1.

The set above is $\{a,b,c\}\circlearrowleft_{\{c\}}$ which is the equivalence class of all renamed variants of $\{a,b,c\}$ under permutations of atoms fixing c (i.e., 'α-abstract a and b in $\{a,b,c\}$). If we apply $[c \mapsto (b,d)]$ to this then we obtain $\{a,c,(b,d)\}\circlearrowleft_{\{b,d\}}$. Note that this is the same set as $\{a,e,(b,d)\}\circlearrowleft_{\{b,d\}}$ and $\{c,e,(b,d)\}\circlearrowleft_{\{b,d\}}$—it is part of our model of α-equivalence on FM that we can 'α-rename abstracted atoms' freely.

Given these ideas, it remains only to decide how substitution interacts with the generalised α-equivalence classes. Full details are in [30, 28].

REFERENCES

[1] Samson Abramsky, Dan R. Ghica, Andrzej S. Murawski, C.-H. Luke Ong, and Ian D. B. Stark, *Nominal games and full abstraction for the nu-calculus*, ***Proceedings of the 19th IEEE symposium on Logic in Computer Science*** (***LICS 2004***), IEEE Computer Society Press, 2004, pp. 150–159.

[2] Brian Aydemir, Aaron Bohannon, and Stephanie Weirich, *Nominal reasoning techniques in Coq*, ***Electronic Notes in Theoretical Computer Science***, vol. 174 (2007), no. 5, pp. 69–77.

[3] Henk P. Barendregt, ***The lambda calculus: its syntax and semantics***, revised ed., North-Holland, 1984.

[4] Stefan Berghofer and Christian Urban, *A head-to-head comparison of de Bruijn indices and names*, ***Electronic Notes in Theoretical Computer Science***, vol. 174 (2007), no. 5, pp. 53–67.

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

[6] NORBERT BRUNNER, *75 years of independence proofs by Fraenkel–Mostowski permutation models*, ***Mathematica Japonica***, vol. 43 (1996), pp. 177–199.

[7] LUÍS CAIRES and LUCA CARDELLI, *A spatial logic for concurrency (part II)*, ***Theoretical Computer Science***, vol. 322 (2004), no. 3, pp. 517–565.

[8] GEORG CANTOR, *Beiträge zur Begründung der transfiniten Mengenlehre II*, ***Mathematische Annalen***, vol. 2 (1897), no. 49, pp. 481–512.

[9] JAMES CHENEY, *A simpler proof theory for nominal logic*, ***Foundations of software science and computation structures*** (Vladimiro Sassone, editor), Lecture Notes in Computer Science, vol. 3441, Springer, 2005, pp. 379–394.

[10] ———, *Completeness and Herbrand theorems for nominal logic*, ***The Journal of Symbolic Logic***, vol. 71 (2006), pp. 299–320.

[11] JAMES CHENEY and CHRISTIAN URBAN, *System description: Alpha-Prolog, a fresh approach to logic programming modulo alpha-equivalence*, Technical Report DSIC-II/12/03, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, 2003.

[12] RANALD CLOUSTON, ***Equational logic for names and binding***, Ph.D. thesis, University of Cambridge, UK, 2010, pending graduation.

[13] PAUL J. COHEN, *The independence of the continuum hypothesis*, ***Proceedings of the National Academy of Sciences of the United States of America***, vol. 50 (1963), no. 6, pp. 1143–1148.

[14] NICOLAAS G. DE BRUIJN, *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem*, ***Indagationes Mathematicae***, vol. 5 (1972), no. 34, pp. 381–392.

[15] JOËLLE DESPEYROUX and ANDRÉ HIRSCHOWITZ, *Higher-order abstract syntax with induction in COQ*, ***Logic for Programming, Artificial intelligence, and Reasoning (LPAR'94)*** (Frank Pfenning, editor), Lecture Notes in Computer Science, vol. 822, Springer, 1994, pp. 159–173.

[16] GILLES DOWEK and MURDOCH J. GABBAY, *Permissive nominal logic*, ***Principles and Practice of Declarative Programming*** (Temur Kutsia, Wolfgang Schreiner, and Maribel Fernández, editors), 2010, pp. 165–176.

[17] HEINZ-DIETER EBBINGHAUS, JÖRG FLUM, and WOLFGANG THOMAS, ***Mathematical logic***, second ed., Springer, 1994.

[18] MARCELO FIORE and DANIELE TURI, *Semantics of name and value passing*, ***Proceedings of the 16th IEEE symposium on Logic in Computer Science (LICS 2001)***, IEEE Computer Society Press, 2001, pp. 93–104.

[19] MARCELO P. FIORE, GORDON D. PLOTKIN, and DANIELE TURI, *Abstract syntax and variable binding*, ***Proceedings of the 14th IEEE symposium on Logic in Computer Science (LICS 1999)***, IEEE Computer Society Press, 1999, pp. 193–202.

[20] MARCELO P. FIORE and SAM STATON, *A congruence rule format for name-passing process calculi from mathematical structural operational semantics*, ***Proceedings of the 21st IEEE symposium on Logic in Computer Science (LICS 2006)***, IEEE Computer Society Press, 2006, pp. 49–58.

[21] ABRAHAM FRAENKEL, *Der Begriff "definit" und die Unabhängigkeit des Auswahlsaxioms*, ***Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse***, 1922, reprinted in English translation in ***From Frege to Gödel: A source book in mathematical logic 1879–1931***, Harvard University Press, second edition, 1971, pp. 253–257.

[22] DOV GABBAY, ***Elementary logics: A procedural perspective***, Prentice Hall, 1998.

[23] MURDOCH J. GABBAY, ***A theory of inductive definitions with alpha-equivalence***, Ph.D. thesis, University of Cambridge, UK, 2000.

[24] ———, *FM-HOL, a higher-order theory of names*, ***35 years of Automath*** (F. Kamareddine, editor), Kluwer, 2002, pp. 247–270.

[25] ———, *Fresh logic*, ***Journal of Applied Logic***, vol. 5 (2007), no. 2, pp. 356–387.

[26] ———, *A general mathematics of names*, ***Information and Computation***, vol. 205 (2007), no. 7, pp. 982–1011.

[27] ———, *Nominal algebra and the HSP theorem*, ***Journal of Logic and Computation***, vol. 19 (2009), no. 2, pp. 341–367.

[28] ———, *A study of substitution, using nominal techniques and Fraenkel–Mostowski sets*, ***Theoretical Computer Science***, vol. 410 (2009), no. 12–13, pp. 1159–1189.

[29] Murdoch J. Gabbay and James Cheney, *A sequent calculus for nominal logic*, ***Proceedings of the 19th IEEE symposium on Logic in Computer Science*** (***LICS 2004***), IEEE Computer Society, 2004, pp. 139–148.

[30] Murdoch J. Gabbay and Michael Gabbay, *Substitution for Fraenkel–Mostowski foundations*, ***Proceedings of the 2008 AISB symposium on computing and philosophy***, 2008, pp. 65–72.

[31] Murdoch J. Gabbay and Martin Hofmann, *Nominal renaming sets*, ***Logic for Programming, Artificial intelligence, and Reasoning*** (***LPAR 2008***), Springer, 2008, pp. 158–173.

[32] Murdoch J. Gabbay and Stéphane Lengrand, *The lambda-context calculus*, ***Electronic Notes in Theoretical Computer Science***, vol. 196 (2008), pp. 19–35.

[33] Murdoch J. Gabbay and Aad Mathijssen, *Capture-avoiding substitution as a nominal algebra*, ***Formal Aspects of Computing***, vol. 20 (2008), no. 4–5, pp. 451–479.

[34] ———, *One-and-a-halfth-order logic*, ***Journal of Logic and Computation***, vol. 18 (2008), no. 4, pp. 521–562.

[35] Murdoch J. Gabbay and Dominic P. Mulligan, *One-and-a-halfth order terms: Curry–Howard for incomplete derivations*, ***Workshop on Logic, Language and Information in Computation*** (***WoLLIC 2008***), Lecture Notes in Artificial Intelligence, vol. 5110, 2008, pp. 180–194.

[36] Murdoch J. Gabbay and Andrew M. Pitts, *A new approach to abstract syntax involving binders*, ***Proceedings of the 14th IEEE symposium on Logic in Computer Science*** (***LICS 1999***), IEEE Computer Society Press, 1999, pp. 214–224.

[37] ———, *A new approach to abstract syntax with variable binding*, ***Formal Aspects of Computing***, vol. 13 (2001), no. 3–5, pp. 341–363.

[38] Martin Hofmann, *Semantical analysis of higher-order abstract syntax*, ***Logic in Computer Science*** (***LICS 1999***), IEEE Computer Society Press, 1999, pp. 204–213.

[39] Furio Honsell, Marino Miculan, and Ivan Scagnetto, *An axiomatic approach to metareasoning on nominal algebras in HOAS*, ***International Colloquium on Automata, Languages and Programming*** (***ICALP 2001***), Lecture Notes in Computer Science, vol. 2076, 2001, pp. 963–978.

[40] Thomas Jech, ***The axiom of choice***, North-Holland, 1973.

[41] ———, ***Set theory***, third ed., Springer, 2006.

[42] Peter T. Johnstone, ***Notes on logic and set theory***, Cambridge University Press, 1987.

[43] ———, ***Sketches of an elephant: A topos theory compendium***, Oxford Logic Guides, vol. 43 and 44, Oxford University Press, 2003.

[44] Edward Keenan and Dag Westerståhl, *Generalized quantifiers in linguistics and logic*, ***Handbook of logic and language*** (J. Van Benthem and A. Ter Meulen, editors), Elsevier, 1996, pp. 837–894.

[45] Jean Louis Krivine and Rene Cori, ***Lambda-calculu, types and models***, Ellis Horwood, 1993, translated by Rene Cori.

[46] Saunders Mac Lane and Ieke Moerdijk, ***Sheaves in geometry and logic: A first introduction to topos theory***, Universitext, Springer, 1992.

[47] MICHAEL MAKKAI, *Avoiding the axiom of choice in general category theory*, ***Journal of Pure and Applied Algebra***, vol. 108 (1996), no. 2, pp. 109–173.

[48] AAD MATHIJSSEN, ***Logical calculi for reasoning with binding***, Ph.D. thesis, Technische Universiteit Eindhoven, 2007.

[49] JAMES MCKINNA and RANDY POLLACK, *Pure Type Systems formalized*, ***Typed Lambda Calculi and Applications*** (***TLCA 1993***) (Marc Bezem and Jan Friso Groote, editors), Lecture Notes in Computer Science, no. 664, Springer-Verlag, 1993, pp. 289–305.

[50] MATIAS MENNI, *About* N*-quantifiers*, ***Applied Categorical Structures***, vol. 11 (2003), no. 5, pp. 421–445.

[51] DALE MILLER and ALWEN TIU, *A proof theory for generic judgments* (extended abstract), ***Proceedings of the 18th IEEE symposium on Logic in Computer Science*** (***LICS 2003***), IEEE Computer Society Press, 2003, pp. 118–127.

[52] ANDRZEJ MOSTOWSKI, *Über die Unabhängigkeit des Wohlordnungssatzes vom Ordnungsprinzip*, ***Fundamenta Mathematicae***, vol. 32 (1939), pp. 201–252.

[53] ———, *On the independence of the well-ordering theorem from the ordering principle*, ***Foundational studies***, Studies in logic and the foundations of mathematics, 93, vol. 1, North-Holland, 1979, pp. 290–338.

[54] DOMINIC P. MULLIGAN, *Online nominal bibliography*, 2010, `http://www.citeulike.org/group/11951/`.

[55] FRANK PFENNING and CONAL ELLIOTT, *Higher-order abstract syntax*, ***PLDI*** (***Programming Language Design and Implementation***), ACM Press, 1988, pp. 199–208.

[56] ANDREW M. PITTS, *Nominal logic, a first order theory of names and binding*, ***Information and Computation***, vol. 186 (2003), no. 2, pp. 165–193.

[57] ———, *Alpha-structural recursion and induction*, ***Journal of the ACM***, vol. 53 (2006), no. 3, pp. 459–506.

[58] ANDREW M. PITTS and MURDOCH J. GABBAY, *A metalanguage for programming with bound names modulo renaming*, ***Mathematics of Program Construction, MPC 2000*** (Roland Carl Backhouse and José Nuno Oliveira, editors), Lecture Notes in Computer Science, vol. 1837, Springer, 2000, pp. 230–255.

[59] MARK R. SHINWELL and ANDREW M. PITTS, *Fresh objective Caml user manual*, Technical Report UCAM-CL-TR-621, University of Cambridge, 2005.

[60] ———, *On a monadic semantics for freshness*, ***Theoretical Computer Science***, vol. 342 (2005), no. 1, pp. 28–55.

[61] MARK R. SHINWELL, ANDREW M. PITTS, and MURDOCH J. GABBAY, *FreshML: Programming with binders made simple*, ***Proceedings of the 8th ACM SIGPLAN International Conference on Functional Programming*** (***ICFP 2003***), vol. 38, ACM Press, 2003, pp. 263–274.

[62] ALLEN STOUGHTON, *Substitution revisited*, ***Theoretical Computer Science***, vol. 59 (1988), no. 3, pp. 317–325.

[63] ALFRED TARSKI and STEVEN GIVANT, ***A formalization of set theory without variables***, vol. 41, American Mathematical Society Colloquium Publications, 1987.

[64] ALWEN TIU, *A logic for reasoning about generic judgments*, ***Electronic Notes in Theoretical Computer Science***, vol. 174 (2007), no. 5, pp. 3–18.

[65] CHRISTIAN URBAN, *Nominal reasoning techniques in Isabelle/HOL*, ***Journal of Automatic Reasoning***, vol. 40 (2008), no. 4, pp. 327–356.

[66] CHRISTIAN URBAN, JAMES CHENEY, and STEFAN BERGHOFER, *Mechanising the metatheory of LF*, ***Proceedings of the 23rd IEEE symposium on Logic in Computer Science*** (***LICS 2008***), IEEE Computer Society Press, 2008, pp. 45–56.

[67] CHRISTIAN URBAN and CHRISTINE TASSON, *Nominal techniques in Isabelle/HOL*, ***Conference on Automated Deduction, Tallinn, Estonia*** (***CADE'05***), Lecture Notes in Artificial Intelligence, vol. 3632, 2005, pp. 38–53.

[68] JOHN VON NEUMANN, *Zur Einführung der transfiniten Zahlen*, ***Acta Universitatis Szegediensis. Acta Scientiarum Mathematicarum***, (1923), no. 1, pp. 199–208, reprinted in English translation in ***From Frege to Gödel: A source book in mathematical logic 1879–1931***, Harvard University Press, second edition, 1971.

[69] DAG WESTERSTÅHL, *Quantifiers in formal and natural languages*, ***Handbook of philosophical logic*** (Gabbay and Günthner, editors), vol. 4, Reidel, 1989, pp. 1–131.

[70] WIKIPEDIA, *Ordinal numbers*, en.wikipedia.org/wiki/Ordinal_number, 23 September 2008.

[71] ERNST ZERMELO, *Über Grenzzahlen und Mengenbereichebe*, ***Fundamenta Mathematicae***, vol. 16 (1930), pp. 29–47.

URL: http://www.gabbay.org.uk