Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 325 (2016) 3–27

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Binding Operators for Nominal Sets

Arthur Azevedo de Amorim[1]

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA, United States

Abstract

The theory of nominal sets is a rich mathematical framework for studying syntax and variable binding. Within it, we can describe several binding disciplines and derive convenient reasoning principles that respect α-equivalence. In this article, we introduce the notion of *binding operator*, a novel construction on nominal sets that unifies and generalizes many forms of binding proposed in the literature. We present general results about these operators, including sufficient conditions for validly using them in inductive definitions of nominal sets.

Keywords: Nominal Sets, Binding, Alpha Equivalence

1 Introduction

Bound variables have puzzled computer scientists and logicians for decades. Although fairly simple to handle in informal pencil-and-paper calculations, they can be surprisingly complex to manage in algorithms and mechanized proofs, where the mostly uninteresting formal details of variable binding cannot be overlooked. Research on the subject has led to various promising approaches for tackling this complexity [6,12,14], among which we can mention the theory of *nominal sets* [5].

Nominal sets constitute a rich mathematical universe where objects contain *variables* that can be *renamed*, allowing various notions of α-equivalence to be defined. In the λ-calculus for example, we stipulate that the term $\lambda x.\, t$ is equivalent to any other obtained by renaming x to a variable y that does not appear free in t, which corresponds to the operation of *name abstraction* on nominal sets [5], used for modeling objects with a single bound variable. The nominal literature has shown how many other forms of binding can be obtained through similar constructions, such

[1] The author thanks Andrew Pitts, Ranald Clouston, and Matthew Weaver for comments on early drafts of this article. This work is supported by NSF award 1513854, *Micro-Policies: A Framework for Tag-Based Security Monitors*.

http://dx.doi.org/10.1016/j.entcs.2016.09.029
1571-0661/© 2016 The Author(s). Published by Elsevier B.V.

as *generalized* name abstractions [4,3], or the binding declarations of Nominal Isabelle [17]. Besides serving as a good theoretical foundation for variable binding, nominal techniques have influenced the design of many tools for manipulating syntax, such as the FreshML programming language [11,15] and the Nominal package for Isabelle/HOL [16,17].

Although some of the notions above are more general than others, none of them proposes to offer a clear, unified picture of what binding means for nominal sets. In this article, we attempt to look at the problem from a more foundational perspective, by introducing *binding operators*: a novel construction on nominal sets that unifies and generalizes many forms of variable binding proposed in the literature. After briefly recalling basic notions of nominal set theory (Section 2), we introduce binding operators in Section 3, showing how to use them for defining a variety of nominal sets representing binders in Section 4. Section 5 gives an alternative characterization of these nominal sets defined by binding operators, used in Section 6 to encompass variable scope within our framework. In Section 7, we discuss category-theoretic properties of binding operators, which provide sufficient conditions for defining nominal sets inductively. We conclude and review related work in Section 8.

2 Preliminaries: Nominal Sets

We begin by recalling basic concepts and results of the theory of nominal sets; for a detailed account on the subject, we refer the reader to the introductory article by Gabbay and Pitts [5] or to Pitts' book [10].

We fix some countably infinite set $\mathbb{A}$. We refer to elements of $\mathbb{A}$ as *atoms*, and use the variable a to denote them. A *permutation* of $\mathbb{A}$ is a bijective function $\pi : \mathbb{A} \to \mathbb{A}$ such that $\pi(a) = a$ for all but finitely many $a \in \mathbb{A}$. Permutations form a group under composition, noted $\mathrm{perm}(\mathbb{A})$; in particular, $\pi \circ \pi' \in \mathrm{perm}(\mathbb{A})$ and $\pi^{-1} \in \mathrm{perm}(\mathbb{A})$ for every $\pi, \pi' \in \mathrm{perm}(\mathbb{A})$.

A *renaming operation* on a set X is a *group action* of $\mathrm{perm}(\mathbb{A})$ on X. Spelled out explicitly, this means a mapping that to each pair $(\pi, x) \in \mathrm{perm}(\mathbb{A}) \times X$ associates an element $\pi \cdot x \in X$, so that

$$1 \cdot x = x \qquad\qquad (\pi_1 \circ \pi_2) \cdot x = \pi_1 \cdot \pi_2 \cdot x,$$

where $1 \in \mathrm{perm}(\mathbb{A})$ denotes the identity function. We treat renaming as right associative, reading $\pi_1 \cdot \pi_2 \cdot x$ as $\pi_1 \cdot (\pi_2 \cdot x)$. The above properties imply in particular $\pi^{-1} \cdot \pi \cdot x = \pi \cdot \pi^{-1} \cdot x = x$ for arbitrary π and x.

We say that a set of atoms A *supports* an element $x \in X$ if the atoms in A completely determine the effect of renaming on x. Formally, if π is a permutation such that $\pi(a) = a$ for every a in A, then $\pi \cdot x = x$. Or, equivalently, if π_1 and π_2 are permutations such that $\pi_1(a) = \pi_2(a)$ for every a in A, then $\pi_1 \cdot x = \pi_2 \cdot x$. If A is finite, we can show [5] that x has a *minimal* finite supporting set $\mathrm{supp}(x)$, by which we mean that $\mathrm{supp}(x)$ is a subset of every finite set A' supporting x. We say that X is a *nominal set* if all of its elements have finite support.

Atoms $\mathbb{A}$ form a nominal set under the action $\pi \cdot a = \pi(a)$, with $\text{supp}(a) = \{a\}$. We can see every set X as a trivial nominal set by posing $\pi \cdot x = x$, which implies $\text{supp}(x) = \emptyset$. We use this structure for sets such as $\mathbb{N}$ or $\mathbb{Z}$, whose elements intuitively do not contain variables. A nominal set with such a trivial renaming operation is called *discrete*. We can also define products and disjoint unions of nominal sets, summarized in the table below.

$X_1 \times X_2$	$\pi \cdot (x_1, x_2) = (\pi \cdot x_1, \pi \cdot x_2)$	$\text{supp}(x_1, x_2) = \text{supp}(x_1) \cup \text{supp}(x_2)$
$X_1 + X_2$	$\pi \cdot (i, x_i) = (i, \pi \cdot x_i)$	$\text{supp}(i, x_i) = \text{supp}(x_i)$

When working with nominal sets, we want to restrict our attention to functions that are well-behaved with respect to renaming. A function $f : X \to Y$ between nominal sets is said to be *equivariant* if it commutes with renaming; that is, $f(\pi \cdot x) = \pi \cdot f(x)$ for every x and π. We write $X \to_{\text{eq}} Y$ for the set of equivariant functions from X to Y. When Y is the discrete nominal set of booleans $\mathbb{B}$, we sometimes say that f is an equivariant property or relation instead. This is equivalent to saying that $f(\pi \cdot x)$ holds if and only if $f(x)$ does. Every such property can be alternatively seen as a *nominal subset* of X; that is, a subset of X that is closed under renaming. Note that equivariant functions cannot add atoms to the support of their arguments: we can show that $\text{supp}(f(x)) \subseteq \text{supp}(x)$, with $\text{supp}(f(x)) = \text{supp}(x)$ if f is injective.

The next best thing to an equivariant function is a *finitely supported* one: a function f between nominal sets X and Y that is *almost* equivariant, *except* for a finite set of atoms A; that is, $f(\pi \cdot x) = \pi \cdot f(x)$ if $\pi(a) = a$ for every $a \in A$. We write $X \to_{\text{fs}} Y$ for the set of finitely supported functions from X to Y. Every equivariant function is trivially finitely supported. Finitely supported functions f form a nominal set under the action $(\pi \cdot f)(x) = \pi \cdot f(\pi^{-1} \cdot x)$. This is equivalent to saying that $(\pi \cdot f)(\pi \cdot x) = \pi \cdot f(x)$ for every π and x, which allows us to depict this renaming operation as acting on a table representation of f:

$$\begin{array}{ccc} x_1 \mapsto f(x_1) & & \pi \cdot x_1 \mapsto \pi \cdot f(x_1) \\ x_2 \mapsto f(x_2) & \xrightarrow{\pi} & \pi \cdot x_2 \mapsto \pi \cdot f(x_2) \\ \vdots & & \vdots \end{array}$$

Note that the support of a function is not computable in general. We use similar actions for other sets of functions; for instance, $\text{perm}(\mathbb{A})$ is a nominal set under the action $\pi \cdot \pi' = \pi \circ \pi' \circ \pi^{-1}$, with $\text{supp}(\pi) = \{a \mid \pi(a) \neq a\}$. Seeing a subset $X' \subseteq X$ as a function $X \to \mathbb{B}$ results in a renaming operation defined by $\pi \cdot X' = \{\pi \cdot x \mid x \in X'\}$.

Let X and Y be two nominal sets, and x and y be elements of X and Y. We say that x and y are fresh with respect to each other, noted $x \# y$, if their supports are disjoint: $\text{supp}(x) \cap \text{supp}(y) = \emptyset$. If $a \in \mathbb{A}$ and $x \in X$, then $a \# x$ simply means that $a \notin \text{supp}(x)$. If $\pi \in \text{perm}(\mathbb{A})$, $\pi \# x$ is equivalent to $\pi(a) = a$ for every $a \in \text{supp}(x)$, which implies $\pi \cdot x = x$. In particular, if f is a finitely supported function, $\pi \# f$ implies $f(\pi \cdot x) = \pi \cdot f(x)$ for every x.

Nominal sets and equivariant functions between them form a category Nom.

It is a complete and cocomplete category; in particular, the initial and terminal objects are the empty and singleton discrete nominal sets, while binary products and sums are given as in the above table. It is also a cartesian closed category, with exponentials given by finitely supported functions.

3 Binding Operators

The most basic form of binding on nominal sets is *name abstraction* [5]. Given a nominal set X, we define an equivalence relation $\equiv_\alpha$ on $\mathbb{A} \times X$ by saying that $(a_1, x_1) \equiv_\alpha (a_2, x_2)$ if and only if

$$\exists a_3.\, a_3 \# (a_1, a_2, x_1, x_2) \wedge (a_1\, a_3) \cdot x_1 = (a_2\, a_3) \cdot x_2 \tag{1}$$

Here, $(a\, a')$ denotes the *transposition* of a and a', which swaps these two atoms while fixing all others. Intuitively, this relation states that a is bound in the pair (a, x) and should be treated up to α-equivalence. If we quotient $\mathbb{A} \times X$ by this relation, we obtain a new nominal set $[\mathbb{A}]X$, called the set of name abstractions of X, where α-equivalent objects become equal.

Besides name abstraction, we can define nominal sets for representing many other binding disciplines, such as name restriction or ML's let rec. A common feature of these constructions is that equivalent elements are obtained by renaming bound atoms while fixing those that remain free. For instance, although not immediately obvious, we can rephrase (1) as

$$(a_1, x_1) \equiv_\alpha (a_2, x_2) \iff \exists \pi.\, \pi \# \operatorname{supp}(x_1) \setminus \{a_1\} \wedge (a_2, x_2) = \pi \cdot (a_1, x_1). \tag{2}$$

Recall that $\pi \# \operatorname{supp}(x_1) \setminus \{a_1\}$ simply means that π fixes all elements of that set. If we interpret the singleton $\{a_1\}$ in this formula as the set of bound variables of the pair (a_1, x_1), we get a generic method for defining α-equivalence for other binders: it suffices to enumerate which atoms should be bound in an object. Formally, we have the following definition.

Definition 3.1 Let X be a nominal set. A *binding operator* on X is an equivariant function $l : X \to_{\text{eq}} \mathcal{P}_{\text{fin}}(\mathbb{A})$. Each l gives rise to a relation $\equiv_l$ on X, defined as

$$x_1 \equiv_l x_2 \iff \exists \pi.\, \pi \# \operatorname{supp}(x_1) \setminus l(x_1) \wedge x_2 = \pi \cdot x_1.$$

Thus, we see that α-equivalence for name abstractions corresponds to a binding operator on $\mathbb{A} \times X$, defined as $l_\alpha(a, x) = \{a\}$. We analyze other examples in Section 4, but need to explain first how exactly binding operators are used to encode binders as nominal sets. Concretely, we show here that every binding operator induces a quotient nominal set, a direct generalization of the analogous results for name abstractions. We begin by noting the following simple facts.

Lemma 3.2 *Let X be a nominal set with a binding operator l. If $x_1 \equiv_l x_2$, then* $\operatorname{supp}(x_1) \setminus l(x_1) = \operatorname{supp}(x_2) \setminus l(x_2)$.

Proof The definition implies that x_2 is of the form $\pi \cdot x_1$, with $\pi \# \operatorname{supp}(x_1) \setminus l(x_1)$; thus, $\pi \cdot (\operatorname{supp}(x_1) \setminus l(x_1)) = \operatorname{supp}(x_1) \setminus l(x_1)$. The result then follows by equivariance, since the right-hand side is equal to $\operatorname{supp}(\pi \cdot x_1) \setminus l(\pi \cdot x_1) = (\pi \cdot \operatorname{supp}(x_1)) \setminus (\pi \cdot l(x_1)) = \pi \cdot (\operatorname{supp}(x_1) \setminus l(x_1))$. □

Lemma 3.3 *Let X be a nominal set endowed with a binding operator l. The $\equiv_l$ relation is an equivariant equivalence relation.*

Proof The relation is clearly reflexive: it suffices to take $\pi = 1$ in its definition. It is also equivariant, because it is defined with equivariant operations.

To see that it is symmetric, take two elements x and x' of X such that $x \equiv_l x'$. By definition, we can find $\pi \in \operatorname{perm}(\mathbb{A})$ such that $\pi \# \operatorname{supp}(x) \setminus l(x)$ and $x' = \pi \cdot x$. We must show that $\pi \cdot x \equiv_l x$. Since $x = \pi^{-1} \cdot \pi \cdot x$, it suffices to show that

$$\pi^{-1} \# \operatorname{supp}(\pi \cdot x) \setminus l(\pi \cdot x),$$

which holds by equivariance, because $\pi^{-1} = \pi \cdot \pi^{-1} = \pi \circ \pi^{-1} \circ \pi^{-1}$.

Finally, let's show transitivity. Take three elements, x_1, x_2 and x_3, such that $x_1 \equiv_l x_2$ and $x_2 \equiv_l x_3$. By unfolding definitions, and using Lemma 3.2, we find π_1 and π_2 such that $x_3 = (\pi_2 \circ \pi_1) \cdot x_1$ and $\pi_i \# (\operatorname{supp}(x_1) \setminus l(x_1))$ for $i = 1, 2$. Since permutation composition is equivariant, we see that $\operatorname{supp}(\pi_2 \circ \pi_1) \subseteq \operatorname{supp}(\pi_1) \cup \operatorname{supp}(\pi_2)$; thus, the freshness conditions above yield $\pi_2 \circ \pi_1 \# \operatorname{supp}(x_1) \setminus l(x_1)$, allowing us to conclude. □

Because binding operators yield equivariant equivalence relations, they lead to quotients that carry a canonical nominal structure:

Lemma 3.4 *Let X be a nominal set with a binding operator l, and let X/l be the quotient of X by the equivalence relation $\equiv_l$. This set possesses a nominal structure satisfying*

$$\pi \cdot [x] = [\pi \cdot x] \qquad\qquad \operatorname{supp}([x]) = \operatorname{supp}(x) \setminus l(x),$$

where $[x]$ denotes the equivalence class of x under $\equiv_l$. In particular, the canonical projection into X/l is equivariant.

Proof Any quotient by an equivariant equivalence relation carries a canonical nominal structure satisfying the first identity [10, Sections 1.8 and 2.9]. We also have [10, Proposition 2.30]

$$\operatorname{supp}([x]) = \bigcap_{x' \equiv_l x} \operatorname{supp}(x'). \tag{3}$$

By Lemma 3.2, the right-hand side equals

$$\operatorname{supp}(x) \setminus l(x) \cup \bigcap_{x' \equiv_l x} l(x').$$

We can conclude because the second term of the union is empty. More precisely, given any atom a in $l(x)$, and any atom a' that is not in $\operatorname{supp}(x)$, we have $(a\,a') \cdot x \equiv_l x$, but $a = (a\,a') \cdot a'$ is not in $l((a\,a') \cdot x)$ by equivariance. □

As a sanity check, if we instantiate the previous result with l_α, the binding operator for name abstractions, we obtain the familiar identities $\pi \cdot \langle a \rangle x = \langle \pi(a) \rangle (\pi \cdot x)$ and $\mathrm{supp}(\langle a \rangle x) = \mathrm{supp}(x) \setminus \{a\}$, where $\langle a \rangle x$ denotes the equivalence class of the pair (a, x) in $[\mathbb{A}]X$. By virtue of being defined as a quotient, we also obtain generic *elimination principles* for such nominal sets. Equivariant functions on them have a particularly simple characterization: they correspond to functions that do not leak bound atoms in their results.

Lemma 3.5 *Let X and Y be nominal sets, and l be a binding operator on X. Let $f : X \to_{\mathrm{eq}} Y$ be a function satisfying $l(x) \# f(x)$ for all x. There exists a unique $\bar{f} : X/l \to_{\mathrm{eq}} Y$ such that $\bar{f}([x]) = f(x)$ for all x. Conversely, every function $f : X \to_{\mathrm{eq}} Y$ that factors through X/l satisfies $l(x) \# f(x)$.*

Proof To build $\bar{f}$, it suffices to show that, for every $x_1 \equiv_l x_2$, we have $f(x_1) = f(x_2)$. By the definition of $\equiv_l$, we find a permutation π that is fresh for $\mathrm{supp}(x_1) \setminus l(x_1)$ such that $x_2 = \pi \cdot x_1$. Thus, $f(x_2) = \pi \cdot f(x_1)$. Since $l(x_1) \# f(x_1)$, it must be the case that $\mathrm{supp}(f(x_1)) \subseteq \mathrm{supp}(x_1) \setminus l(x_1)$. This implies that $\pi \# f(x_1)$, and thus $f(x_1) = \pi \cdot f(x_1) = f(x_2)$. The last assertion follows because, if $f(x) = \bar{f}([x])$ for some equivariant function $\bar{f}$, then $\mathrm{supp}(f(x))$ is contained in $\mathrm{supp}([x])$, which equals $\mathrm{supp}(x) \setminus l(x)$. □

Later, in Lemma 6.14, we extend this result to describe the finitely supported functions that can be defined on such quotients. It would be possible (and not too difficult) to state this extension right away and prove it directly, but the tools developed in Section 6 provide more structure for attacking the problem.

Notice that the above properties only rely on knowing which atoms are bound in an object, and how these atoms are affected by renaming. This is indeed the only piece of information that we can extract from binding operators, which hide everything else that we might care about in bound atoms—for instance, the order in which they appear. Fortunately, as shown in the rest of this paper, this extra information is irrelevant for deriving the fundamental properties of binding constructs in nominal sets.

4 Examples

Given the generality of binding operators, it is worth analyzing a few examples of binding disciplines that they can express. We show here how to model a few syntactic constructs that have been extensively studied in the literature. We include an example of common idiom that is not directly supported by our framework—namely, binding atoms in only *part* of an object. Fortunately, as shown in Section 6, this limitation is not fundamental, and can be overcome by adding a notion of scope to binding operators.

4.1 Generalized Name Abstraction

Name abstraction binds a single atom within an object. The simplest way to generalize it is to consider constructions where bound atoms are specified by arbitrary data structures that contain atoms, leading to so-called *generalized* name abstractions [4]. Specifically, given nominal sets X and Y, we adapt the binding operator l_α defining name abstraction to $X \times Y$, by setting $l_\alpha(x, y) = \mathrm{supp}(x)$. The corresponding quotient, noted $[X]Y$, is known as the nominal set of X-abstractions of Y.

4.2 Name Restriction

Processes in the π-calculus [8] communicate through named channels, which can be made private using a form of binding known as *name restriction*. Concretely, an expression of the form $\nu a.\, t$ denotes a computation t that has access to a communication channel a bound by ν, which cannot be used by any other processes defined outside of this expression.

Besides being subject to α-equivalence of bound channel names, π-calculus processes satisfy certain behavioral identities related to name restriction. For instance, the order in which channels are bound in a process expression is irrelevant:

$$\nu a_1.\, \nu a_2.\, t = \nu a_2.\, \nu a_1.\, t.$$

To represent name restriction, we could be tempted to model π-calculus terms with a nominal set of the form $[\mathcal{P}_{\mathrm{fin}}(\mathbb{A})]E$, where E denotes a nominal set of process expressions. The idea is that an expression of the form $\nu a_1.\, \ldots\, \nu a_n.\, t$ would correspond to the element $[(\{a_1, \ldots, a_n\}, t)]$. Unfortunately, this encoding does not validate another basic property of name restriction: spurious private channels do not affect process behavior. Formally, if a does not occur free in t, then $\nu a.\, t = t$.

We solve this problem by restricting our binding operator to a nominal subset of $\mathcal{P}_{\mathrm{fin}}(\mathbb{A}) \times E$ that excludes spurious atoms. Specifically, we pose

$$L(E) = \{(A, t) \in \mathcal{P}_{\mathrm{fin}}(\mathbb{A}) \times E \mid A \subseteq \mathrm{supp}(t)\},$$

and quotient this nominal set by the binding operator $l(A, t) \triangleq A$. The resulting nominal set, noted $\mathrm{Res}(E)$, is known as the *free nominal restriction set* over E [10, Chapter 9]. We can then represent an expression $\nu a_1.\, \ldots\, \nu a_n.\, t$ by the element $[(\{a_1, \ldots, a_n\} \cap \mathrm{supp}(t), t)]$. Besides their application to the π-calculus, free nominal restriction sets yield a monad on Nom that provides an useful model of fresh-name generation.

4.3 Mutually Recursive Definitions

Most programming languages allow mutually recursive function definitions. In the ML family, these usually take the form

$$\mathsf{let\ rec}\ a_1 = t_1\ \mathsf{and}\ \cdots\ \mathsf{and}\ a_n = t_n\ \mathsf{in}\ t,$$

where the atoms $a_1, \dots, a_n$ are bound in the expressions $t, t_1, \dots, t_n$. We could represent mutually recursive definitions with a nominal set of the form $[\mathrm{List}(\mathbb{A})]\,\mathrm{List}(E)$, where E is some nominal set of expressions, and $\mathrm{List}(X)$ is the nominal set of finite lists of elements of X. The idea is that an expression such as the one above would be mapped to the element $[([a_1, \dots, a_n], [t_1, \dots, t_n, t])]$. The problem, as noted by Pottier [12], is that this nominal set contains elements that do not correspond to any valid expression, such as $[([a], [])]$, where the number of defined atoms does not match the number of definition bodies.

We can use binding operators to model mutually recursive definitions by viewing expressions as the one above as a pair (f, t), where $t \in E$ is an expression, and $f : \mathbb{A} \rightharpoonup_{\mathrm{fin}} E$ is a partial function with finite domain. The term t represents the result of the expression, while the function f maps each atom to the corresponding definition; in the example given above, this would be a function mapping a_1 to t_1, a_2 to t_2, etc. (Note that this assumes that the order of the definitions does not matter. We could also have considered a more concrete variant with lists of declarations that have an explicit order.) The bound atoms in (f, t) are exactly those in the domain of f, which leads to the nominal set $\mathrm{Mut}(E) \triangleq ((\mathbb{A} \rightharpoonup_{\mathrm{fin}} E) \times E)/(\mathrm{dom} \circ p_1)$, where p_1 designates the first projection function. This solution is similar to others proposed in the literature [12,17].

4.4 An Obstacle: Binder Scope

We could try to adapt the previous example to model parallel nonrecursive definitions:

$$\mathsf{let}\ a_1 = t_1\ \mathsf{and}\ \cdots\ \mathsf{and}\ a_n = t_n\ \mathsf{in}\ t,$$

where the atoms $a_1, \dots, a_n$ are bound in t, but not in $t_1, \dots, t_n$. Unfortunately, the tools that we have developed so far cannot take this form of scoped binding into account, because the equivalence derived from binding operators require atoms to be renamed *everywhere*, including in positions where they should remain free (in this case, the t_i). We will see later, in Section 6, how to work around this issue by considering renaming operations that act only on a limited scope within an object.

5 Binding Functions

It is basic set theory that every surjective function f corresponds to a quotient by an equivalence relation—namely, the one where x_1 and x_2 are equivalent if and only if $f(x_1) = f(x_2)$. In this section, we prove an analogous result for binding operators, showing that their quotients can alternatively be characterized as what we call *binding functions*. This characterization will be useful in Section 6, where we use it to relate a more general class of quotients on nominal sets to binding operators.

Definition 5.1 An equivariant function f between nominal sets is a *binding function* if it is surjective and, whenever $f(x_1) = f(x_2)$, we have $x_1 \equiv_{l_f} x_2$, where

$l_f(x) = \mathrm{supp}(x) \setminus \mathrm{supp}(f(x))$.[2] This last condition simply means that there exists a permutation π, with $\pi \# f(x_1)$, such that $x_2 = \pi \cdot x_1$.

Intuitively, a binding function is one that removes atoms from the support of its argument, but doesn't discard any other information attached to it. Notice that, by construction, the projection into a quotient by a binding operator is a binding function. However, the converse also holds: every binding function corresponds to a quotient by a binding operator. More precisely, we have the following result.

Lemma 5.2 *If $f : X \to_{\mathrm{eq}} Y$ is a binding function, then there is an isomorphism $i : Y \cong X/l_f$ such that $i(f(x)) = [x]$. In other words, f is a coequalizer of the equivalence relation $\equiv_{l_f}$.*

Proof We already know that f is surjective and that $f(x_1) = f(x_2)$ implies $x_1 \equiv_{l_f} x_2$. Conversely, we can see that $x_1 \equiv_{l_f} x_2$ implies $f(x_1) = f(x_2)$, since $f(x_1) = f(\pi \cdot x_1)$ when $\pi \# f(x_1)$. This proves that f is the coequalizer we're looking for.□

Binding operators can be combined by composing their quotients:

Lemma 5.3 *If f and g are binding, then so is $h = gf$, and $l_h(x) = l_f(x) \cup l_g(f(x))$.*

Proof It is clear that h is surjective, as the composition of two surjections. Now, suppose that $g(f(x_1)) = g(f(x_2))$. Since g is binding, we find a permutation π, with $\pi \# g(f(x_1))$, such that $f(x_2) = \pi \cdot f(x_1) = f(\pi \cdot x_1)$. But f is also binding, so we get another permutation π' such that $\pi' \# f(\pi \cdot x_1)$ and $x_2 = \pi' \cdot \pi \cdot x_1$. The freshness assumptions on π and π' imply that $\pi' \circ \pi \# g(f(x_1))$, because

$$\mathrm{supp}(g(f(x_1))) = \mathrm{supp}(g(f(\pi \cdot x_1))) \subseteq \mathrm{supp}(f(\pi \cdot x_1)).$$

This shows that h is binding. The last claim follows because

$$\begin{aligned} l_h(x) &= \mathrm{supp}(x) \setminus \mathrm{supp}(g(f(x))) \\ &= (\mathrm{supp}(x) \setminus \mathrm{supp}(f(x))) \cup (\mathrm{supp}(f(x)) \setminus \mathrm{supp}(g(f(x)))) \\ &= l_f(x) \cup l_g(f(x)). \end{aligned}$$

□

Finally, as an aside, we note that every equivariant function can be factored through a quotient by a binding operator. We can compare this result to the more familiar one that says that every function can be factored through its image.

Lemma 5.4 *Let $f : X \to_{\mathrm{eq}} Y$ be an equivariant function. We can factor f through X/l_f as $f = \bar{f} \circ [-]$:*

$$\begin{tikzcd} X \arrow[r, two heads] & X/l_f \arrow[r, "\bar{f}"] & Y, \end{tikzcd}$$

[2] Kurz et al. [7, Notation 5.40] refer to this binding operator as the set of "bound variables relative to a map", and use it to study variable binding in infinite objects.

where $\bar{f}$ preserves supports, *in the following sense:*

$$\mathrm{supp}(\bar{f}(\bar{x})) = \mathrm{supp}(\bar{x}).$$

Furthermore, this factorization is unique up to isomorphism. Specifically, if $f = hg$, where g is binding and h preserves supports, there exists an isomorphism i such that the following diagram commutes:

$$\begin{tikzcd} X \arrow[r, two heads] \arrow[d, two heads, "g"] & X/l_f \arrow[d, "\bar{f}"] \\ Y' \arrow[r, "h"] \arrow[ur, equal, "i"] & Y \end{tikzcd}$$

Proof That f can be factored through X/l_f follows from Lemma 3.5. Because h preserves supports, we find that $l_g = l_f$, and construct i using Lemma 5.2. □

The analogy with the image of a function goes even further: we can show that binding functions and functions that preserve supports form a *factorization system* [1, Definition 5.5.1] on the category of nominal sets. Spelled out in detail, this means that both classes of functions contain all isomorphisms, are closed under composition, and can be used to factor uniquely (up to isomorphism) any equivariant function, as shown above.

6 Atom Scope and Freshening

In Section 4.4, we noted that binding operators cannot express syntactic forms that bind atoms in only part of an object. We show here how to accommodate such constructs by decomposing the renaming operation of a nominal set into smaller independent ones. The idea is that each of these independent operations applies an atom permutation to part of an object without affecting the rest, thus allowing bound atoms to α-vary within their intended scope. The corresponding quotients are *not* binding functions in the sense of Definition 5.1, but we show here that they still support similar elimination principles to those obtained from Lemma 3.5. Besides allowing us to model more binders, this machinery will be useful for deriving stronger elimination principles that work with finitely supported functions.

6.1 Independence

The main technical device that we need is the notion of *independence* of two renaming operations.

Definition 6.1 Let X be a set with two renaming operations, $\odot_1$ and $\odot_2$. We say that these operations are *independent* if they commute, in the following sense:

$$\pi_1 \odot_1 \pi_2 \odot_2 x = \pi_2 \odot_2 \pi_1 \odot_1 x.$$

We use the $\odot$ operator to denote a set of multiple independent renaming operations on a set, whereas $\cdot$ is reserved to its canonical nominal structure. If the elements of

X are finitely supported with respect to these operations, we say that X has two independent nominal structures. We note $\mathrm{supp}_1(x)$ and $\mathrm{supp}_2(x)$ the supports of an element x of X with respect to each of the renaming operations.

As an example, if X and Y are nominal sets, we can define two independent renaming operations on the product $X \times Y$ by posing

$$\pi \odot_1 (x, y) \triangleq (\pi \cdot x, y) \qquad\qquad \pi \odot_2 (x, y) \triangleq (x, \pi \cdot y).$$

Note that we can express the product nominal set $X \times Y$ as the composition of these two operations. As a matter of fact, any set with two independent renaming operations can be endowed with a compound one, as shown in the following results.

Lemma 6.2 *Let X be a set with two independent nominal structures. The support of an element with respect to one structure is invariant with respect to the other:*

$$\mathrm{supp}_1(\pi \odot_2 x) = \mathrm{supp}_1(x) \qquad\qquad \mathrm{supp}_2(\pi \odot_1 x) = \mathrm{supp}_2(x).$$

Proof We only need to show one case, the other one following analogously. Given two atoms a and a', we have

$$\left(a\, a'\right) \odot_1 \pi \odot_2 x = \pi \odot_2 \left(a\, a'\right) \odot_1 x.$$

Since renaming operations are injective, we see that $(a\, a') \odot_1 x = x$ if and only if $(a\, a') \odot_1 \pi \odot_2 x = \pi \odot_2 x$. But a is in $\mathrm{supp}_1(x)$ if and only if there are infinitely many a' such that $(a\, a') \odot_1 x \neq x$, and similarly for $\pi \odot_2 x$. This allows us to conclude.□

Lemma 6.3 *Let X be a set with two independent nominal structures. We can define a compound nominal structure on X by setting*

$$\pi \cdot x \triangleq \pi \odot_1 \pi \odot_2 x.$$

Each of the $\odot_i$ is equivariant with respect to this compound operation, in the following sense:

$$\pi \cdot \pi' \odot_i x = (\pi \cdot \pi') \odot_i \pi \cdot x.$$

Finally, the support of an element is the union of the supports of the constituent parts:

$$\mathrm{supp}(x) = \mathrm{supp}_1(x) \cup \mathrm{supp}_2(x).$$

Proof By unpacking definitions, we can check directly that this compound operation indeed satisfies the required properties of a renaming operation, and that each $\odot_i$ is equivariant. We can also see that every element is finitely supported: given x in X and a permutation π such that $\pi \# \mathrm{supp}_1(x) \cup \mathrm{supp}_2(x)$, we have

$$\pi \cdot x = \pi \odot_1 \pi \odot_2 x = \pi \odot_1 x = x,$$

proving that $\mathrm{supp}(x)$ exists and that it is contained in $\mathrm{supp}_1(x) \cup \mathrm{supp}_2(x)$. Note that $\mathrm{supp}_1(x) \cup \mathrm{supp}_2(x)$ depends equivariantly on x, thanks to Lemma 6.2:

$$\begin{aligned}
&\mathrm{supp}_1(\pi \cdot x) \cup \mathrm{supp}_2(\pi \cdot x) \\
&= \mathrm{supp}_1(\pi \odot_1 \pi \odot_2 x) \cup \mathrm{supp}_2(\pi \odot_2 \pi \odot_1 x) \\
&= \pi \cdot \mathrm{supp}_1(\pi \odot_2 x) \cup \pi \cdot \mathrm{supp}_2(\pi \odot_1 x) \\
&= \pi \cdot \mathrm{supp}_1(x) \cup \pi \cdot \mathrm{supp}_2(x) \\
&= \pi \cdot (\mathrm{supp}_1(x) \cup \mathrm{supp}_2(x)).
\end{aligned}$$

Thus, $\mathrm{supp}_1(x) \cup \mathrm{supp}_2(x)$ is also contained in $\mathrm{supp}(x)$, which proves that both sets are equal. □

By iterating this process, we can combine any finite number of independent renaming operations. For simplicity, we restrict ourselves to the case of two independent operations in what follows, but the theory developed here can be generalized without difficulty to the case of a finite number of renaming operations that are pairwise independent. Whenever a set has multiple nominal structures, we consider the compound one defined in the above lemma as canonical.

6.2 Local Equivariance and Binding Operators

If a nominal set can be decomposed into independent renaming operations, we can express the scope of a binder on that set by instantiating the generic notion of α-equivalence in Definition 3.1 to a particular renaming operation. However, if we want the corresponding quotient to behave nicely with respect to the "global" nominal structure, we must require that the corresponding binding operator be independent of the other renaming operations. This leads to *local* variants of the notions of equivariance and binding operator.

Definition 6.4 Let X be a set with two independent nominal structures, and Y be a nominal set. We say that a function f is *locally equivariant* (with respect to $\odot_i$) if

$$f(\pi \odot_j x) = \begin{cases} \pi \cdot f(x) & \text{if } j = i \\ f(x) & \text{otherwise} \end{cases}.$$

We note the set of such functions as $X \to^i_{\mathrm{eq}} Y$.

By Lemma 6.2, we see that supp_i is locally equivariant with respect to the corresponding nominal structure. Furthermore:

Lemma 6.5 *Using the same notations as above, a function $f : X \to^i_{\mathrm{eq}} Y$ is also equivariant with respect to the compound nominal structure of Lemma 6.3.*

Proof Assuming $i = 1$, we have $f(\pi \cdot x) = f(\pi \odot_1 \pi \odot_2 x) = \pi \cdot f(\pi \odot_2 x) = \pi \cdot f(x)$. The other case is similar. □

Definition 6.6 Let X be a set with two independent nominal structures. A *local binding operator* (with respect to $\odot_i$) is a locally equivariant function $l : X \to^i_{\mathrm{eq}} \mathcal{P}_{\mathrm{fin}}(\mathbb{A})$.

By Lemma 6.5, a local binding operator l for a renaming operation $\odot_i$ is a binding operator for two different nominal structures, and thus gives rise to two different notions of α-equivalence. To distinguish between them, we use $x_1 \equiv_l x_2$ to say that x_1 and x_2 are α-equivalent with respect to the compound structure, and $x_1 \equiv^i_l x_2$ to say that x_1 and x_2 are α-equivalent with respect to $\odot_i$. If we unfold the definition of α-equivalence for the last case, it simply means that there exists a permutation π fixing $\mathrm{supp}_i(x) \setminus l(x)$ such that $x_2 = \pi \odot_i x_1$. Its corresponding quotient is compatible with all the nominal structures of the original set, as shown below.

Lemma 6.7 *Let $l : X \to^i_{\mathrm{eq}} \mathcal{P}_{\mathrm{fin}}(\mathbb{A})$ be a local binding operator. The relation $\equiv^i_l$ is equivariant with respect to the operations $\odot_j$ and with respect to $\cdot$. The quotient $X/\equiv^i_l$ has the following nominal structures:*

$$\pi \odot_j [x] = [\pi \odot_j x] \qquad \mathrm{supp}_j([x]) = \begin{cases} \mathrm{supp}_i(x) \setminus l(x) & \text{if } j = i \\ \mathrm{supp}_j(x) & \text{otherwise} \end{cases}$$

$$\pi \cdot [x] = [\pi \cdot x] \qquad \mathrm{supp}([x]) = \mathrm{supp}_i(x) \setminus l(x) \cup \mathrm{supp}_{i'}(x),$$

where $i' \neq i$ in the last equation. In particular, the renaming operations $\odot_j$ are independent.

Proof We assume $i = 1$, the other case being symmetric. It suffices to show the result for $\odot_1$ and $\odot_2$, since these two cases combined yield the results for $\cdot$. Let's start with equivariance. We already know that $\equiv^1_l$ is equivariant with respect to $\odot_1$ from Lemma 3.3. To show that it is also the case for $\odot_2$, suppose that we have a permutation π such that $\pi \# \mathrm{supp}_1(x) \setminus l(x)$, so that $x \equiv^1_l \pi \odot_1 x$. We want to show that, for any permutation π', we have

$$\pi' \odot_2 x \equiv^1_l \pi' \odot_2 \pi \odot_1 x = \pi \odot_1 \pi' \odot_2 x.$$

This holds because, by local equivariance, π is fresh for $\mathrm{supp}_1(\pi' \odot_2 x) \setminus l(\pi' \odot_2 x) = \mathrm{supp}_1(x) \setminus l(x)$.

Finally, the definition of the renaming operations on $X/\equiv^1_l$, and their independence, follow from equivariance. We already know how to compute $\mathrm{supp}_1([x])$ from the earlier Lemma 3.4. Thus, to conclude, we just have to compute $\mathrm{supp}_2([x])$. But $x_1 \equiv^1_l x_2$ implies $\mathrm{supp}_2(x_1) = \mathrm{supp}_2(x_2)$ by Lemma 6.2, and thus $\mathrm{supp}_2([x]) = \mathrm{supp}_2(x)$ (cf. (3) in the proof of Lemma 3.4). □

To understand how this construction works, let's revisit the example of parallel definitions of Section 4.4. Once again, if E is some nominal set of program expressions, we model raw parallel definitions (that is, before taking the quotient) with the nominal set $(\mathbb{A} \rightharpoonup_{\mathrm{fin}} E) \times E$. We can decompose this nominal structure into two independent ones defined as

$$\pi \odot_1 (f, e) \triangleq (f \circ \pi^{-1}, \pi \cdot e) \qquad \pi \odot_2 (f, e) \triangleq (a \mapsto \pi \cdot f(a), e).$$

Thus, in a let expression

$$\mathsf{let}\ a_1 = t_1\ \mathsf{and}\ \cdots\ \mathsf{and}\ a_n = t_n\ \mathsf{in}\ t,$$

the operation $\odot_1$ renames the atoms on the left-hand side of the definitions, as well as the ones in t, whereas $\odot_2$ only renames those that occur in the t_i. We see that the binding operator $l(f, e) \triangleq \mathrm{dom}(f)$ is local to $\odot_1$, and lists precisely the atoms on the left-hand side of the local definitions. Unlike the case of mutually recursive definitions discussed in Section 4.3, the definition of $\equiv_l^1$ guarantees that the bound atoms of a pair (f, e) cannot vary in the bodies of local definitions in f. Thus, we can represent parallel let with the nominal set $\mathrm{Par}(E) = ((\mathbb{A} \rightharpoonup_{\mathrm{fin}} E) \times E)/\equiv_l^1$.

6.3 *Elimination Principles*

Now that we have quotient nominal sets that correspond to local binding operators, we turn our attention to the functions that can be defined on them. If we want a function that is locally equivariant with respect to the same nominal structure as the local binding operator that we considering, it suffices to apply Lemma 3.5 directly. More generally, however, we want to define functions that are *not* locally equivariant, but only equivariant with respect the compound nominal structure. Going back to the example of parallel let, the function $c(f, e) = |\mathrm{supp}(f, e)|$ that counts the number of variables in an expression is not locally equivariant, since renaming parts of an expression independently may change its result. For instance, the expressions

$$\mathsf{let}\, a_1 = a_1\, \mathsf{in}\, a_1$$

and

$$\mathsf{let}\, a_1 = a_2\, \mathsf{in}\, a_1$$

have a different number of variables, but can be obtained from each other by a local renaming.

We cannot describe these functions using the compact elimination principle of Lemma 3.5, since, as stated earlier, projecting into such a quotient is not a binding function. This can be seen, for instance, in the identity $\mathrm{supp}([x]) = \mathrm{supp}_1(x) \setminus l(x) \cup \mathrm{supp}_2(x)$ of Lemma 6.7, which implies in particular that an atom a may appear in the support of $[x]$ even if it occurs in $l(x)$. As it turns out, we can express the quotient by a local binding operator on X as a quotient by a "global" binding operator on a nominal subset of X, where bound atoms are prevented from aliasing the ones that remain free after the quotient by α-equivalence. Specifically, we now prove that $X/\equiv_l^1$ is isomorphic to the quotient $X_{\#l}/l$, where $l : X \to_{\mathrm{eq}}^1 \mathcal{P}_{\mathrm{fin}}(\mathbb{A})$ is a local binding operator, and

$$X_{\#l} = \{x \in X \mid l(x) \# \mathrm{supp}_2(x)\}. \tag{4}$$

(Note that $X_{\#l}$ is a nominal subset of X for its compound nominal structure, but not for any of the $\odot_i$.) In particular, using Lemma 3.5, this allows us to define equivariant functions $X/{\equiv_l^1} \to_{\mathrm{eq}} Y$ (with respect to the compound nominal structure of $X/{\equiv_l^1}$) through equivariant functions $f : X_{\#l} \to_{\mathrm{eq}} Y$ that satisfy $l(x) \# f(x)$ for any x in $X_{\#l}$. We begin with the following results, showing how to avoid conflicting with sets of "bad" atoms when choosing concrete values for the ones that are bound.

Lemma 6.8 *Let X be a nominal set with a binding operator l. Given $\bar{x} \in X/l$ and a finite set of atoms A, we can find $x \in X$ such that $[x] = \bar{x}$ and $l(x) \# A$.*

Proof Pick any representative x' of $\bar{x}$. We cannot choose x to be x' right away, since in principle the set $l(x')$ may not be fresh with respect to A. We can, however, rename the conflicting atoms to fresh values.

Choose a set of atoms A' such that $|A'| = |l(x') \cap A|$ and $A' \# (x', A)$. By a cardinality argument, we can construct a permutation π that sends $l(x') \cap A$ to A' while leaving all other atoms fixed. By construction, π does not affect the free atoms of x'; formally, $\mathrm{supp}(\pi) = l(x') \cap A \cup A'$, hence $\pi \# \mathrm{supp}(x') \setminus l(x')$. This implies that $[\pi \cdot x'] = [x'] = \bar{x}$. We then can choose x to be $\pi \cdot x'$, provided that we show that its atoms are completely fresh (that is, $\pi \cdot l(x') \# A$). The result follows because the definition of π implies that $\pi \cdot l(x') = A' \cup l(x') \setminus A$, and both parts are disjoint from A. □

Lemma 6.9 *Let X be a nominal set with a binding operator l, and A a finite set of atoms. Let x_1 and x_2 be two elements of X such that $x_1 \equiv_l x_2$, $l(x_1) \# A$, and $l(x_2) \# A$. There exists a permutation π such that $\pi \# A$, $\pi \# \mathrm{supp}(x_1) \setminus l(x_1)$, and $x_2 = \pi \cdot x_1$.*

Proof By the definition of $\equiv_l$, we can find some permutation π' that is fresh for $\mathrm{supp}(x_1) \setminus l(x_1)$, and such that $x_2 = \pi' \cdot x_1$. By basic properties of permutations, there exists a permutation π such that

$$\begin{aligned} \pi(a) &= \pi'(a) && \text{if } a \in l(x_1) \\ \mathrm{supp}(\pi) &\subseteq l(x_1) \cup l(x_2). \end{aligned}$$

The set $l(x_1) \cup l(x_2)$ is disjoint from $\mathrm{supp}(x_1) \setminus l(x_1)$ and A, implying that π is fresh for $\mathrm{supp}(x_1) \setminus l(x_1)$ and A. In order to conclude, it suffices to show that $\pi \cdot x_1 = \pi' \cdot x_1$, which holds because π and π' agree on $\mathrm{supp}(x_1)$. □

We can now explain how local binding operators yield binding functions.

Lemma 6.10 *Let X be a set with two independent nominal structures, and l a local binding operator on X with respect to the operation $\odot_1$. There exists an isomorphism $X/{\equiv_l^1} \cong X_{\#l}/l$ making the following diagram commute:*

$$\begin{tikzcd} X \arrow[r, two heads] & X/{\equiv_l^1} \\ X_{\#l} \arrow[u, hook] \arrow[r, two heads] & X_{\#l}/l \arrow[u, equal] \end{tikzcd}$$

In particular, to build an equivariant function $\bar{f} : X/{\equiv_l^1} \to_{\text{eq}} Y$, it suffices to find an equivariant $f : X_{\#l} \to_{\text{eq}} Y$ such that $l(x) \# f(x)$ for every x; then, $\bar{f}([x]) = f(x)$ whenever x is in $X_{\#l}$.

Proof Consider the canonical projection into $X/{\equiv_l^1}$ restricted to the nominal subset $X_{\#l}$. Call this function f. By Lemma 5.2, it suffices to show that f is binding, and that its corresponding binding operator, l_f, is equal to l. The last point follows by unfolding definitions and making use of the freshness constraints on the elements of $X_{\#l}$. Moreover, we can show that f is surjective using Lemma 6.8. The only part that is missing is showing that $f(x_1) = f(x_2)$ implies $x_1 \equiv_l x_2$ for any x_1 and x_2 in $X_{\#l}$. Note that $f(x_1) = f(x_2)$ is equivalent to $x_1 \equiv_l^1 x_2$. Using Lemma 6.9, we find a permutation π that is fresh for $\text{supp}_2(x_1)$ and $\text{supp}_1(x_1) \setminus l(x_1)$ such that $x_2 = \pi \odot_1 x_1$. We must then show that $x \equiv_l \pi \odot_1 x$. The assumptions on π imply that

$$x = \pi \odot_2 x \tag{5}$$
$$\pi \# \text{supp}_1(x) \setminus l(x) \cup \text{supp}_2(x). \tag{6}$$

Thus, showing $x \equiv_l \pi \odot_1 x$ is tantamount to showing $x \equiv_l \pi \cdot x = \pi \odot_1 \pi \odot_2 x$. We conclude using (6), which, given that $l(x) \# \text{supp}_2(x)$, is equivalent to $\pi \# \text{supp}(x) \setminus l(x)$, □

By applying this result to the nominal set $\text{Par}(E)$, and unfolding definitions, we find that it is isomorphic to

$$\{(f, e) \in (\mathbb{A} \rightharpoonup_{\text{fin}} E) \times E \mid \text{dom}(f) \# \text{im}(f)\}/(\text{dom} \circ p_1),$$

where p_1 is the first projection. This shows that even if we can construct elements of $\text{Par}(E)$ by giving a let expression where some of the free atoms in the bodies of the local definitions are shadowed, when defining functions on that set, we can assume that the locally defined atoms are disjoint from the ones that are free. We can see this fact as a restatement, in nominal terms, of Barendregt's well-known variable convention, of which Lemma 6.10 is a formal justification.

6.4 Multiple Quotients

Although the above results have been stated for quotients by a single local binding operator, they can also be composed to derive elimination principles for quotients by multiple operators. For this, we can make use of the following simple observation, which allows us to combine the freshness constraints arising from multiple quotients.

Lemma 6.11 *Let X be a nominal set with a binding operator l. Every nominal subset $\bar{X}$ of X/l is of the form*

$$\{x \in X \mid [x] \in \bar{X}\}/l.$$

Proof By Lemma 5.2. □

As an example of elimination principle for multiple quotients, we have the following result.

Lemma 6.12 *Let X be a set with two independent nominal structures and two local binding operators, $l_1 : X \to^1_{\mathrm{eq}} \mathcal{P}_{\mathrm{fin}}(\mathbb{A})$ and $l_2 : X \to^2_{\mathrm{eq}} \mathcal{P}_{\mathrm{fin}}(\mathbb{A})$. Let $\equiv_{l_1,l_2}$ be the composition of the equivalence relations $\equiv^1_{l_1}$ and $\equiv^2_{l_2}$. There is an isomorphism $X/{\equiv_{l_1,l_2}} \cong X_{\#l_1,l_2}/(l_1 \cup l_2)$ such that the following diagram commutes:*

$$\begin{tikzcd}
X \arrow[r, two heads] & X/{\equiv_{l_1,l_2}} \arrow[d, equal] \\
X_{\#l_1,l_2} \arrow[u, hook] \arrow[r, two heads] & X_{\#l_1,l_2}/(l_1 \cup l_2)
\end{tikzcd}$$

where

$$X_{\#l_1,l_2} = \{x \in X \mid l_1(x) \# \mathrm{supp}_2(x), l_2(x) \# \mathrm{supp}_1(x)\}.$$

Proof Because both renaming operations are independent, the composition $\equiv_{l_1,l_2}$ is indeed an (equivariant) equivalence relation. We can express the quotient by this equivalence relation as an iterated quotient, which, thanks to Lemma 6.10, has the form

$$X/{\equiv_{l_1,l_2}} \cong (X_{\#l_1}/l_1)_{\#\bar{l}_2}/\bar{l}_2,$$

where $\bar{l}_2$ denotes the lifting of the binding operator l_2 to $X_{\#l_1}/l_1$, using Lemma 3.5. By Lemma 6.11, we have

$$(X_{\#l_1}/l_1)_{\#\bar{l}_2} \cong \{x \in X_{\#l_1} \mid l_2(x) \# \mathrm{supp}_1(x) \setminus l_1(x)\}/l_1 \tag{7}$$

Since $l_2(x) \subseteq \mathrm{supp}_2(x)$ for every $x \in X$, we can see that $l_1(x) \# l_2(x)$ when $x \in X_{\#l_1}$. Thus, the right-hand side of (7) is precisely $X_{\#l_1,l_2}/l_1$. We conclude using the fact that

$$X_{\#l_1,l_2}/l_1/\bar{l}_2 \cong X_{\#l_1,l_2}/(l_1 \cup l_2),$$

thanks to Lemma 5.3. It is a tedious but straightforward exercise to check that the composition of these isomorphims results in the above commuting diagram. □

It is worth spelling out explicitly a special case of this result.

Lemma 6.13 *Let X, Y be nominal sets and l_X, l_Y be binding operators over them. Define the* separated product *to be the following nominal subset of $X \times Y$:*

$$(X, l_X) \otimes (Y, l_Y) \triangleq \{(x, y) \in X \times Y \mid x \# l_Y(y), l_X(x) \# y\}$$

Let $l(x, y) \triangleq l_X(x) \cup l_Y(y)$. There is an isomorphism

$$\sigma : X/l_X \times Y/l_Y \cong ((X, l_X) \otimes (Y, l_Y))/l$$

satisfying $\sigma([x],[y]) = [(x,y)]$ *for all* $(x,y) \in (X, l_X) \otimes (Y, l_Y)$.

Proof As pointed out before, we can define two independent renaming structures on $X \times Y$:

$$\pi \odot_1 (x,y) = (\pi \cdot x, y) \qquad\qquad \pi \odot_2 (x,y) = (x, \pi \cdot y).$$

We can check that l_X and l_Y can be lifted to local binding operators l_1 and l_2 on $X \times Y$, and that the separated product $(X, l_X) \otimes (Y, l_Y)$ is just $(X \times Y)_{\#l_1,l_2}$, as defined in Lemma 6.12. We conclude by noting that the product relation $\equiv_{l_X} \times \equiv_{l_Y}$ is the composition of $\equiv^1_{l_1}$ and $\equiv^2_{l_2}$, and that

$$X/l_X \times Y/l_Y \cong (X \times Y)/(\equiv_{l_X} \times \equiv_{l_Y}) \cong (X \times Y)_{\#l_1,l_2}/l,$$

where the last isomorphism follows from Lemma 6.12. □

With this result, we can finally state a strong elimination principle for binding operators, which describes finitely supported functions defined on their quotients.

Lemma 6.14 *Let* X *and* Y *be nominal sets,* l *be a binding operator on* X*, and* $f : X \to_{\text{fs}} Y$ *a finitely supported function that satisfies the following* freshness condition for binders*: if* x *is such that* $l_X(x) \# f$*, then* $l_X(x) \# f(x)$*. There exists* $\bar{f} : X/l_X \to_{\text{fs}} Y$ *satisfying* $\bar{f}([x]) = f(x)$ *for all* x *such that* $l_X(x) \# f$.

Proof Roughly, we can use the previous result to express $\bar{f}$ as the partial application of a suitable evaluation function. Define the nominal set

$$F \triangleq \{g : X \to_{\text{fs}} Y \mid \forall x.\, l_X(x) \# g \Rightarrow l_X(x) \# g(x)\}.$$

Pose $l_F(g) \triangleq \emptyset$ and $l(g,x) \triangleq l_X(x)$. Let $P \triangleq (F, l_F) \otimes (X, l_X)$ and $e : P \to_{\text{eq}} Y$ be the evaluation function $e(g,x) \triangleq g(x)$. By construction, e satisfies $l(g,x) \# e(g,x)$ for every $(g,x) \in P$. Using Lemma 3.5, we can thus construct $\bar{e} : P/l \to_{\text{eq}} Y$ such that $\bar{e}([(g,x)]) = g(x)$. We then pose $\bar{f}(\bar{x}) \triangleq \bar{e}(\sigma([f],\bar{x}))$, where σ is the isomorphism of Lemma 6.13. □

Unlike the case for equivariant functions, the mapping $f \mapsto \bar{f}$ defined above is *not* bijective in general, because it only uses part of the information contained in its argument: in order to have $\bar{f} = \bar{g}$, we just have to guarantee that $f(x) = g(x)$ for all x such that $l_X(x) \# (f,g)$.

Lemma 6.14 is the analog for binding operators of an earlier result on name abstractions [9], which says that we can obtain a function $\bar{f} : [\mathbb{A}]X \to_{\text{fs}} Y$ by finding $f : \mathbb{A} \times X \to_{\text{fs}} Y$ satisfying $a \# f(a,x)$ when $a \# f$—exactly what we obtain by instantiating our result with the operator l_α defining name abstraction.

7 Functorial Properties

So far, we have used binding operators to define individual syntactic constructs, but still have not determined when such constructs can be combined into valid complete grammars. Previous results show that this is possible in many cases, such as grammars given by *nominal signatures* [18]. This allows us for instance to define the set of λ-terms modulo α-equivalence as the solution of the equation

$$\Lambda = \mathbb{A} + \Lambda^2 + [\mathbb{A}]\Lambda, \tag{8}$$

which says that a λ-term is either a variable, a pair of λ-terms representing an application, or the name abstraction of a λ-term, representing a function definition.

In this section, we extend these results to a large class of nominal sets defined with binding operators. It is standard to interpret definitions such as the one above as specifying the initial algebra of a certain functor. What makes the definition of Λ valid is that name abstraction can be made into a functor, and that this functor satisfies certain technical conditions needed for the construction of initial algebras. Thus, we want to determine which endofunctors on Nom can be defined through binding operators and to study their properties. The first step is to recast some of the earlier definitions into a more structured form, showing that the process of quotienting by a binding operator is itself functorial.

Definition 7.1 The category of binding operators Bnd is defined as follows. Objects are pairs (X, l_X), where X is a nominal set and l_X is a binding operator over X. When no ambiguity can arise, we use X to refer to the pair (X, l_X), and we sometimes omit the X index from l_X. A morphism from X to Y is an equivariant function $f : X \to_{\mathrm{eq}} Y$ such that, for every x in X,

$$l_Y(f(x)) = l_X(x) \cap \mathrm{supp}(f(x)).$$

We note $U : \mathsf{Bnd} \to \mathsf{Nom}$ the obvious forgetful functor that maps (X, l_X) to X.

Intuitively, this definition says that applying a morphism f in Bnd to an argument x cannot change the status of the atoms in $\mathrm{supp}(x)$ from bound to free, or vice versa. Note, however, that applying f may still *remove* atoms from the support of x entirely, both bound and free. This restriction guarantees that every such f can be lifted to quotients, as shown in the following result.

Lemma 7.2 *We can extend quotients by binding operators to a functor $Q : \mathsf{Bnd} \to \mathsf{Nom}$ satisfying*

$$Q(X) = X/l_X \qquad\qquad Q(f)([x]) = [f(x)].$$

Note that the second identity says that the canonical projections form a natural transformation $U \to Q$. This functor has a right adjoint $Z : \mathsf{Nom} \to \mathsf{Bnd}$, which associates to a nominal set X the constant binding operator $l(x) = \emptyset$. Furthermore, the QZ is naturally isomorphic to the identity on Nom, via the canonical projection into the quotient.

Proof We define the action of Q on morphisms by appealing to Lemma 3.5. Specifically, let X and Y be two objects in Bnd, and $f : X \to Y$ be a morphism between them. We know that f and $[-]$ are equivariant, thus it suffices to show that

$l_X(x) \# [f(x)]$ for all x in X. Since $\mathrm{supp}([f(x)]) = \mathrm{supp}(f(x)) \setminus l_Y(f(x))$, this is equivalent to

$$l_X(x) \cap \mathrm{supp}(f(x)) \setminus l_Y(f(x)) = \emptyset,$$

which readily follows from the fact that f is a morphism in Bnd. It is easy to verify that this construction preserves identities and composition; thus, Q is indeed a functor.

To show that Z is right adjoint to Q, consider an equivariant function $f : Q(X) \to_{\mathrm{eq}} Y$, where $X \in \mathsf{Bnd}$ and Y is a nominal set. The composite $g = f \circ [-]$ is an equivariant function that satisfies $l_X(x) \# g(x)$ for every x in X. Thus, g is a morphism $X \to Z(Y)$ in Bnd. Conversely, given a morphism $g : X \to Z(Y)$ in Bnd, we can use Lemma 3.5 to factor it as $f \circ [-]$, with $f : Q(X) \to Y$. We can readily check that these constructions are mutally inverse, and natural in X and Y. The last assertion follows from Lemma 5.2. □

We note that the condition on morphisms of Definition 7.1 is not tight, in the sense that the above proof would still work with the weaker assumption

$$l_Y(f(x)) \supseteq l_X(x) \cap \mathrm{supp}(f(x)),$$

which intuitively says that f may bind atoms that are free in x. The reason for choosing the stronger variant, as we will see, is that it allows us to characterize Bnd as a category of coalgebras, which will play an important role later on, when studying functors involving quotients.

To define a functor on Nom via binding operators, we can define a functor $F : \mathsf{Nom} \to \mathsf{Bnd}$, and then consider the composite QF. It is easy to see that the examples discussed so far—name abstractions, name restriction, mutually recursive and parallel definitions—can be extended into functors by following this recipe. For name abstractions, for instance, we can take $F(X)$ to be $\mathbb{A} \times X$, endowed with the binding operator $l_\alpha(a, x) = \{a\}$, which can be extended to a functor in the obvious way.

7.1 Strengthening Quotients

Many functors derived from binding operators allow arbitrary finitely supported functions to be lifted, not just equivariant ones. A good example is name abstraction: given any finitely supported function $f : X \to_{\mathrm{fs}} Y$, we can define $[\mathbb{A}]f : [\mathbb{A}]X \to_{\mathrm{fs}} [\mathbb{A}]Y$ satisfying $[\mathbb{A}]f(\langle a \rangle x) = \langle a \rangle(f(x))$ whenever $a \# f$. In category-theory jargon, such functors are known as *enriched*.

Formally, to enrich a functor $G : \mathsf{Nom} \to \mathsf{Nom}$ means to extend its action on morphisms to a family of equivariant functions $(X \to_{\mathrm{fs}} Y) \to_{\mathrm{eq}} (G(X) \to_{\mathrm{fs}} G(Y))$ satisfying the usual functor laws. An equivariant action is compatible with the structure of Nom, which allows us to generalize properties of G to finitely supported functions. For instance, if G has an initial algebra, it supports a recursion scheme that for defining finitely supported functions.

If G is of the form QF, it can be enriched by appealing to the elimination principle of Lemma 6.14, but the quotient structure provides a more direct route.

Indeed, it is well-known that enriching QF is equivalent to giving it a *strength*: a natural transformation $\eta_{X,Y} : X \times QF(Y) \to QF(X \times Y)$ satisfying the laws depicted below.

$$\begin{tikzcd}
1 \times QF(A) \arrow[r, "\eta"] \arrow[dr] & QF(1 \times A) \arrow[d] \\
& QF(A)
\end{tikzcd}
\qquad
\begin{tikzcd}
(A \times B) \times QF(C) \arrow[r, "\eta"] \arrow[d] & QF((A \times B) \times C) \arrow[dd] \\
A \times (B \times QF(C)) \arrow[d, "A\times\eta"] & \\
A \times QF(B \times C) \arrow[r, "\eta"] & QF(A \times (B \times C))
\end{tikzcd}$$

Intuitively, η allows us to lift functions by currying the composite

$$(X \to_{\text{fs}} Y) \times QF(X) \xrightarrow{\eta} QF((X \to_{\text{fs}} Y) \times X) \xrightarrow{F(\epsilon)} QF(Y) \ ,$$

where ϵ denotes the evaluation function $(X \to_{\text{fs}} Y) \times X \to_{\text{eq}} Y$. The strength laws then guarantee that the resulting action satisfies the required functor laws.

Now, note that the separated product $\otimes$ of Lemma 6.13 admits a trivial extension into a bifunctor $\mathsf{Bnd}^2 \to \mathsf{Bnd}$, endowing Bnd with the structure of a symmetric monoidal category, with unit $Z(1)$; furthermore, its isomorphism $\sigma : Q(X)\times Q(Y) \cong Q(X \otimes Y)$ is natural in X and Y, and satisfies all laws required to make Q a strong monoidal functor from $(\mathsf{Bnd}, \otimes, Z(1))$ to $(\mathsf{Nom}, \times, 1)$. This allows us to strengthen QF by composition: it suffices to find a natural transformation $\eta'_{X,Y} : Z(X) \otimes F(Y) \to F(X \times Y)$ in Bnd satisfying laws analogous to the ones above. It is then a simple exercise to check that the composite

$$X \times QF(Y) \xrightarrow{\cong} QZ(X) \times QF(Y) \xrightarrow{\cong} Q(Z(X) \otimes F(Y)) \xrightarrow{Q\eta'} QF(X \times Y)$$

is a strength on QF. Indeed, all of the functors arising from binding operators studied here can be trivially strengthened in this fashion.

7.2 Preservation of Colimits and Initial Algebras

After analyzing the matter of strength, let's turn our attention to other properties of quotient functors—namely, which colimits they preserve. Among other things, this is useful for building initial algebras. The initial algebra of a functor $G : \mathsf{Nom} \to \mathsf{Nom}$ is normally constructed as the colimit of the chain diagram

$$\emptyset \xrightarrow{\iota} G(\emptyset) \xrightarrow{G(\iota)} G^2(\emptyset) \xrightarrow{G^2(\iota)} \cdots \ ,$$

but this construction only makes sense if G preserves that colimit, which can often be reduced to showing that the individual functors appearing in definition of G preserve colimits of the same shape.

Note that, since Q has a right adjoint, QF preserves all colimits that are preserved by F. But F takes values in a category of binding operators, which must in principle be taken into account when computing these colimits. We show here is that this is not the case: we can always reduce colimits in Bnd to simpler colimits in

Nom, by characterizing the former as the Eilenberg-Moore category of the following comonad.

Lemma 7.3 *The L construction used in Section 4.2 for modeling name restriction can be extended into a functor* Nom $\to$ Nom *by setting*

$$L(f)(A, x) = (A \cap \mathrm{supp}(f(x)), f(x))$$

This functor has the structure of a comonad, given by natural transformations $\rho : L \to 1$ and $\nu : L \to L^2$ defined by

$$\rho(A, x) = x \qquad\qquad \nu(A, x) = (A, (A, x))$$

and satisfying the usual conditions: $L\nu \circ \nu = \nu L \circ \nu$ and $L\rho \circ \nu = \rho L \circ \nu = 1_L$.

Proof It is easy to check that the action of L on morphisms is functorial; for instance,

$$\begin{aligned} L(f)(L(g)(A, x)) &= (A \cap \mathrm{supp}(g(x)) \cap \mathrm{supp}(f(g(x))), f(g(x))) \\ &= (A \cap \mathrm{supp}(f(g(x))), f(g(x))) \\ &= L(f \circ g)(x), \end{aligned}$$

where we made use of the fact that $\mathrm{supp}(f(g(x))) \subseteq \mathrm{supp}(g(x))$. Checking that (L, ρ, ν) forms a comonad is similarly straightforward. □

Theorem 7.4 *The category* Bnd *is equivalent to the Eilenberg-Moore category of coalgebras of the comonad (L, ρ, ν). We recall that objects of this category are pairs (X, l) of a nominal set X and a map $l : X \to_{\mathrm{eq}} L(X)$ satisfying the first two laws depicted below. A morphism from (X, l_X) to (Y, l_Y) is an equivariant function $f : X \to_{\mathrm{eq}} Y$ making the third diagram below commute.*

$$\begin{tikzcd} X \arrow[r, "l"] \arrow[dr, "1"'] & L(X) \arrow[d, "\rho"] \\ & X \end{tikzcd} \quad \begin{tikzcd} X \arrow[r, "l"] \arrow[d, "l"] & L(X) \arrow[d, "\nu"] \\ L(X) \arrow[r, "L(l)"] & L^2(X) \end{tikzcd} \quad \begin{tikzcd} X \arrow[r, "f"] \arrow[d, "l_X"] & Y \arrow[d, "l_Y"] \\ L(X) \arrow[r, "L(f)"] & L(Y) \end{tikzcd}$$

Proof An equivariant function $l : X \to_{\mathrm{eq}} L(X)$ satisfying the commuting triangle above is of the form $l(x) = (l'(x), x)$, proving that such a function is equivalent to a binding operator on X. The first commuting square is valid for any l satisfying the triangle. The second commuting square is just a restatement of the restriction imposed on morphisms in Bnd. □

As with every Eilenberg-Moore category, we obtain a right adjoint $\bar{L}$ to the forgetful functor U : Bnd $\to$ Nom. By unpacking the definitions, we can see this right adjoint as endowing each $L(X)$ with a binding operator $l(A, x) \triangleq A$, exactly what we used to model name restriction in Section 4.2. We now have all the required ingredients to prove the main result of this section.

Theorem 7.5 *Let $\mathcal{I}$ be a small category and $D : \mathcal{I} \to \mathsf{Bnd}$ a diagram. Suppose that $(U(\rho_i) : U(D_i) \to U(C))_{i\in\mathcal{I}}$ is a colimiting cocone in Nom. Then so is $(Q(\rho_i) : Q(D_i) \to Q(C))_{i\in\mathcal{I}}$. In particular, any functor of the form QF preserves all colimits that are preserved by UF.*

Proof As previously noted, since Q has a right adjoint, it suffices to show that $(\rho_i : D_i \to C)$ is a colimiting cocone in Bnd. But this holds because U creates colimits, thanks to general results on Eilenberg-Moore categories [2, Props. 4.1.4 and 4.3.1]. □

Remark 7.6 The preceding result does not hold for limits in general. For a counterexample, consider the functor $S : \mathsf{Nom} \to \mathsf{Bnd}$, defined as $S(X) = (X, \mathrm{supp})$, with the obvious action on morphisms. We have (trivially) that $US(\mathbb{A}^2) = US(\mathbb{A})^2 = \mathbb{A}^2$. On the other hand, $QS(\mathbb{A}^2)$ has two elements ($[(a,a)]$ and $[(a,a')]$, where $a \neq a'$), whereas the product $QS(\mathbb{A})^2$ has only one.

As mentioned previously, one application of Theorem 7.5 is showing that a functor of the form QF can be used for defining grammars by initial algebras, which often follows from simple category-theoretic reasons. For instance, suppose that $F = (X \times (-), l_\alpha)$ is the functor defining generalized name abstractions $[X](-)$, as in Section 4. Then $UF = X \times (-)$, which preserves all colimits because Nom is cartesian closed.

Another potential application is providing sufficient conditions for the existence of right adjoints of quotient functors. One of the many corollaries of the adjoint functor theorem says that, for a functor $\mathsf{Nom} \to \mathsf{Nom}$, preserving arbitrary colimits and having a right adjoint are equivalent, because Nom is a Grothendieck topos. If that functor is of the form QF, then Theorem 7.5 allows us to check only whether UF preserves arbitrary colimits. We immediately conclude, for the same reasons as before, that generalized name abstractions have a right adjoint. Although this particular right adjoint already had a good explicit characterization [3], we think that our construction helps shed light on the relation between binding and adjunctions.

8 Conclusion and Related Work

Binding operators are an expressive framework for defining binders for nominal sets, encompassing many constructs that have been previously proposed in the literature. Although it is not clear how much expressive power our operators add compared to previous approaches, we believe that they provide a uniform, concise explanation for many of the properties enjoyed by binding constructs, such as their elimination principles, functoriality, compatibility with inductive definitions, etc.

Since the early development of nominal sets, researchers have directed their attention to more general forms of binding than name abstraction. The simplest such construction is given by generalized name abstractions, studied by Gabbay [4] and others. Clouston [3] investigated some of their categorical properties, in particular the related notion of *separating function*, and adjunctions between generalized name abstractions and the so-called *freshening function space*. That work provides an explicit construction of this adjunction, which could be interesting to generalize to other quotients by binding operators.

The Nominal Isabelle package features a rich language for defining data types with binders [17], allowing users to specify which atoms are bound in values of a data type. Unlike the present work, their focus is not in offering a foundational definition of binding, but in providing a usable and flexible tool. One point of similarity is that they use a general class of functions to enumerate which atoms are bound. Although such functions are more limited than general binding operators, the mechanism is rich enough to capture interesting binders, including generalized name abstractions and free nominal restriction sets. One way in which Nominal Isabelle goes beyond our framework is by allowing two parts of a term to be renamed independently, and yet share the same set of bound atoms. For instance, assuming that we have two different atoms a and a', this would allow to equate terms of the form

$$\mathsf{Foo}\ \{a, a'\}\ (a, a')\ (a, a') = \mathsf{Foo}\ \{a, a'\}\ (a, a')\ (a', a),$$

assuming that the definition of Foo is such that it binds the set $\{a, a'\}$ separately on its second and third arguments; that is, we allow swapping the a and a' in the third argument independently of the second.

To our knowledge, the closest relative of binding operators and their quotients is the operation of *simple monoidal sum* studied by Schöpp [13, Section 3.3.2]; we quickly review that construction here, adapted to our conventions and notations. We start with an arbitrary equivariant function $f : X \to_{\mathrm{eq}} Y * A$, where $Y * A$ denotes the "full" separated product, in which both components are not allowed to share any atoms. We then define a binding operator l on X by posing $l(x) = \mathrm{supp}(p_2(f(x)))$. By construction, $l(x)\ \#\ p_1(f(x))$ for every x. Thus, we can lift $g = p_1 \circ f$ to $\bar{g} : X/l \to_{\mathrm{eq}} Y$, which we call the simple monoidal sum of f. Viewed this way, this construction is a small generalization of quotients by binding operators; indeed, we can recover the latter by taking A to be $\mathcal{P}_{\mathrm{fin}}(\mathbb{A})$. The main difference between both works is that Schöpp uses simple monoidal sums to interpret a form of dependent sum in a nominal type theory, studying properties of that construction that are more relevant in that context, whereas we attempt to provide a more elementary presentation of binding, quotients, and their properties.

References

[1] F. Borceux. *Handbook of Categorical Algebra: Volume 1, Basic Category Theory.* Cambridge Textbooks in Linguistics. Cambridge University Press, 1994.

[2] F. Borceux. *Handbook of Categorical Algebra: Volume 2, Categories and Structures.* Cambridge Studies in Philosophy. Cambridge University Press, 1994.

[3] R. Clouston. Generalised name abstraction for nominal sets. In *FoSSaCS*, volume 7794 of *Lecture Notes in Computer Science*, pages 434–449. Springer, 2013.

[4] M. J. Gabbay. FM-HOL, a higher-order theory of names. In *In Thirty Five years of Automath, Heriot-Watt*, 2002.

[5] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13(3-5):341–363, 2002.

[6] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, Jan. 1993.

[7] A. Kurz, D. Petrişan, P. Severi, and F. de Vries. Nominal coalgebraic data types with applications to lambda calculus. *Logical Methods in Computer Science*, 9(4), 2013.

[8] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I. *Inf. Comput.*, 100(1):1–40, Sept. 1992.

[9] A. M. Pitts. Alpha-structural recursion and induction. *J. ACM*, 53(3):459–506, May 2006.

[10] A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, NY, USA, 2013.

[11] A. M. Pitts and M. J. Gabbay. A metalanguage for programming with bound names modulo renaming. In *Mathematics of Program Construction, volume 1837 of Lecture Notes in Computer Science*, pages 230–255. Springer-Verlag, 2000.

[12] F. Pottier. An overview of Cαml. *Electron. Notes Theor. Comput. Sci.*, 148(2):27–52, Mar. 2006.

[13] U. Schöpp. *Names and Binding in Type Theory*. PhD thesis, University of Edinburgh, 2006.

[14] P. Sewell, F. Z. Nardelli, S. Owens, G. Peskine, T. Ridge, S. Sarkar, and R. Strnisa. Ott: Effective tool support for the working semanticist. *J. Funct. Program.*, 20(1):71–122, 2010.

[15] M. R. Shinwell, A. M. Pitts, and M. J. Gabbay. FreshML: Programming with binders made simple. In *Proceedings of the 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003)*, volume 38, pages 263–274. ACM Press, August 2003.

[16] C. Urban. Nominal techniques in Isabelle/HOL. *J. Autom. Reason.*, 40(4):327–356, May 2008.

[17] C. Urban and C. Kaliszyk. General bindings and alpha-equivalence in nominal Isabelle. *Logical Methods in Computer Science*, 8(2), 2012.

[18] C. Urban, A. Pitts, and M. Gabbay. Nominal unification. In *Computer Science Logic: 17th International Workshop CSL 2003, 12th Annual Conference of the EACSL, 8th Kurt Gödel Colloquium, KGC 2003, Vienna, Austria, August 25-30, 2003. Proceedings*, pages 513–527, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.