Supported Sets – A New Foundation For Nominal Sets And Automata

Thorsten Wißmann
Institute for Computing and Information Sciences, Radboud University
Nijmegen, the Netherlands

Abstract

This proposes and discusses the category of supported sets which provides a uniform foundation for nominal sets of various kinds, such as those for equality symmetry, for the order symmetry, and renaming sets. We show that all these differently flavoured categories of nominal sets are monadic over supported sets. Thus, supported sets provide a canonical finite way to represent nominal sets and the automata therein, e.g. register automata. Name binding in supported sets is modelled by a functor following the idea of de Bruijn indices. This functor lifts to the well-known abstraction functor in nominal sets. Together with the monadicity result, this gives rise to a determinization process that takes the finite representation of a register automaton in supported sets and transforms it into its configuration automaton in nominal sets.

ACM Reference Format:
Thorsten Wißmann. 2022. Supported Sets – A New Foundation For Nominal Sets And Automata. Manuscript, Submitted (January 2022), 22 pages.

Nominal sets provide an elegant framework to reason about structures that involve names, the permutation of names, and name binding. Originally used 100 years ago by Fraenkel and Mostowski for the entirely different purpose of axiomatic set theory, Gabbay and Pitts [15] re-discovered them in 1999 to define λ-expressions modulo α-equivalence as an inductive data type. Its associated structural recursion principle allowed it to define capture avoiding substitution as a total function. Since then, a plethora of results using nominal sets were established in many areas, including proof assistants [5, 32], calculi [14, 31], and automata models [7, 22, 29]. Nominal automata are capable of processing words over infinite alphabets (*data alphabets*), while having good computational properties. Their expressiveness is similar (and sometimes identical) to that of register automata [8, 11, 21], which are automata with a *finite* description processing infinite alphabets. This finiteness condition translates into the notion of *orbit-finiteness* in the nominal world, which requires extra work to obtain a finite description of nominal automata, because orbit-finite objects are infinite in general.

In recent years, more general concepts of nominal sets were considered that generalize from the permutation of names to other operations. One branch called *renaming sets* [17, 26, 31] allows that distinct names are mapped to the same identifier; in another branch, symmetries on other data alphabets [7, 34] were considered, for example monotone bijections on rational numbers $\mathbb{Q}$, called the *total order symmetry*.

Nominal sets are not the only categorical approach to name binding. Multiple presheaf-based approaches to name binding were developed over the years [9, 12, 13] which are strongly related to nominal sets [19]. In contrast to nominal sets, the name binding in presheaf approaches follow the method of *de Bruijn indices* [10].

Despite their rich categorical structure, it is known that the category of nominal sets (in all above-mentioned flavours) is not *monadic* over sets, that is, their theory is not an algebraic theory that can be described by a monad on sets.

In order to still benefit from monadicity, we introduce the category of *supported sets* and show that nominal sets are monadic over those. Thus, we make nominal sets applicable to results and methods known from algebraic categories, such as the generalized powerset construction [6, 30] or results about representation of algebras [1] that provide a canonical finite representation of orbit-finite nominal sets. Moreover, supported sets do not require symmetries on the data alphabet, so they can also serve as a categorical foundation for data alphabets that are not described by symmetries. In fact, we discovered supported sets while working on a learning algorithm for register automata for general data alphabets.

Structure of the paper. We start by a high-level overview of the contributions of the present paper (Section 1), before recalling the basic definitions for nominal sets (Section 2). After introducing supported sets and discussing its basic categorical properties (Section 3), we show that nominal sets are monadic over it (Section 4). As applications of the monadicity, we show that supported sets have a functor for name binding that lifts to nominal sets (Section 5), and that we have a construction from automata in supported sets to nominal automata (Section 6). After comparing supported sets to presheaf categories (Section 7), we conclude the paper.

Author's address: Thorsten Wißmann, Institute for Computing and Information Sciences, Radboud University, Nijmegen, the Netherlands.

2022. /2022/1-ART

1 Overview of the Contribution

Let us first take a more higher-level look at the properties and benefits of supported sets. The proposed category has a very simple definition, and so we state it already here in full detail. For a fixed set A, the category of *supported sets* $\mathsf{Supp}(A)$ contains the following data:

A supported set (X, s_X) is a set X equipped with a map $\mathsf{s}_X\colon X \to \mathcal{P}_\mathsf{f}(A)$ (where $\mathcal{P}_\mathsf{f}$ denotes finite powerset).

A supported map $f\colon (X, \mathsf{s}_X) \to (Y, \mathsf{s}_Y)$ is a map $f\colon X \to Y$ with $\mathsf{s}_Y(f(x)) \subseteq \mathsf{s}_X(x)$ for all $x \in X$.

This definition reflects the basic principles when working with data alphabets A:

1. The terms or structures $x \in X$ of interest store finitely many data values $\mathsf{s}_X(x) \subseteq A$.
2. The computations f on such a structure x can not invent new data values out of thin air, but may decide to drop data values, e.g. by clearing a register: $\mathsf{s}_Y(f(x)) \subseteq \mathsf{s}_X(x)$.

Inspired by nominal set nomenclature, $\mathsf{s}_X(x)$ is called *the support of* x. However, the big difference to nominal sets is that the map s_X is a structural property of a supported set X and not a derived notion. Thus, we can define supported sets by simply specifying X and s_X without any notion of permutation or renaming of data values.

Properties. $\mathsf{Supp}(A)$ has nice categorical properties:

1. $\mathsf{Supp}(A)$ is locally finite, that is, a supported set is *finitely presentable* iff it is finite.
2. $\mathsf{Supp}(A)$ is complete, cocomplete, and cartesian closed; the forgetful functor to sets preserves all colimits and all finite limits.

Binding. Despite the little structure of supported sets, we define *name binding* as a functor on $\mathsf{Supp}(A)$ (for countably infinite A) that is reminiscent of de Bruijn indices [10]. Thus, the binding is similar to that in presheaf approaches [12], but different to the permutation based *abstraction functor* of nominal sets. We show that the binding functor of $\mathsf{Supp}(A)$ in fact *lifts* to the category of nominal sets (for equality symmetry) and that it is (naturally isomorphic to) the abstraction functor.

Monadicity. The different *nominal theories* (with their different operations) can respectively be modelled as a monad on supported sets. If A is a countable infinite set, the following categories are then *monadic* over $\mathsf{Supp}(A)$:

1. nominal sets for the equality symmetry (operations are permutations on the data alphabet A) [15, 28].
2. nominal renaming sets (operations are maps that are allowed to identify elements of A) [17, 26, 31].
3. nominal sets for the total order symmetry (the data alphabet is the set of rational numbers $\mathbb{Q}$ and the operations are monotone bijections) [7, 34].

These monadic adjunctions make a couple of general results about monadic adjunctions applicable:

Finite representation [1]. Orbit-finite nominal set are infinite in general. But by the monadicity, a nominal set is orbit-finite iff it can be described by a finite supported set of *generators* and finitely many *equations*.

Determinization [6, 30]. Register automata can be modelled as finite coalgebras in supported sets. Their state space is simply a finite supported set and their transition structure reads the input data symbols and reassigns the register contents. This reassignment can be considered a side effect with respect to the monad modelling the nominal theory of interest. The generalized powerset construction [30] then internalizes these side effects, and yields an automaton without side effects in the Eilenberg-Moore category of the monad, that is, an automaton in nominal sets. This process effectively turns the finite register automaton into the orbit-finite automaton on the configurations (similarly to the classical powerset construction, where the state space of the resulting deterministic automaton is the configuration space of the original non-deterministic automaton).

This process of determinization shows that supported sets should not be understood as a competitor to nominal sets, but as a common foundation for different nominal symmetries. And moreover, it may serve as a categorical framework for data alphabets that do not admit symmetries at all.

2 Preliminaries on Nominal Sets

Before discussing the categorical relationship between nominal sets and supported sets in detail, we first recall some notations and basic concepts of nominal sets. We assume that the reader is familiar with basic category theory, but we restrict to set-theoretic definitions wherever possible.

Notation 2.1. Given sets X, Y, we write Y^X for the set of all maps $X \to Y$. Injective maps are denoted by $\rightarrowtail$, surjective maps by $\twoheadrightarrow$. We write $\cdot$ for the composition of maps. We fix sets $1 = \{0\}$, $2 = \{0, 1\}$.

We use the usual notation for cycles to define bijective maps, i.e. given a set A and elements $a_0, \ldots a_{n-1}$, the notation

$$(a_0\ a_1 \cdots a_{n-1})$$

denotes the bijective map $f\colon A \to A$ given by

$$f(x) = \begin{cases} a_{k+1 \bmod n} & \text{if there is some } k < n \text{ with } x = a_k \\ x & \text{otherwise.} \end{cases}$$

Nominal sets and various flavours thereof are built around the notion of monoid actions, which specifies how atoms nested in some structure can be permuted or renamed.

Definition 2.2. Given a monoid $(M, \cdot, e)$, an M-set is a set X together with a map '$\cdot$': $M \times X \to X$, called the *action* and written infix $m \cdot x$ for $x \in X$, $m \in M$, such that $e \cdot x = x$ and $(m \cdot m') \cdot x = m \cdot (m' \cdot x)$ for all $m, m' \in M$, $x \in X$. Thus, we use '$\cdot$' both for the monoid multiplication and the action. A map $f\colon X \to Y$ between M-sets $(X,\cdot)$, $(Y,\star)$ is *equivariant* if $f(m\cdot x) = m\star f(x)$ for all $m \in M, x \in X, y \in Y$. The category of M-sets and equivariant maps is denoted by M-Set.

Remark 2.3. Equivalently, an M-set can be defined a set X together with a monoid homomorphism $M \to (X^X, \cdot, \mathsf{id}_X)$, where X^X is the monoid of maps on X, with composition as the monoid multiplication.

Throughout the paper, M will be a submonoid of $(A^A, \cdot, \mathsf{id}_A)$ for a set A, written $M \le A^A$, which most of the results are parametric in. The set A is called the set of *atoms*, which can intuitively be understood as names of registers or data-values that appear in the input of an automaton or as names used for binding operations. The submonoid M determines a subset of maps of interest, that contains the identity and is closed under composition. Thus, we use $\cdot$ both for map composition and monoid-multiplication, and moreover, the unit of the monoid M is simply id_A.

An element $x \in X$ of an M-set $(X,\cdot)$ can be understood as some structure with elements from A embedded, for example, λ-expressions whose variable names are elements of A, or states in a register automaton that keep track of characters from an input alphabet A. For $f\colon A \to A$, $f \in M$, and $x \in X$, the element $f \cdot x \in X$ is intuitively obtained by replacing every a in x with $f(a)$.

Example 2.4. For every set A and $M \le A^A$, the following sets are M-sets:

1. The set A itself with the action $m \cdot a := m(a)$, for $m \in M, a \in A$.
2. If X is an M-set, then so is $\mathcal{P}_\mathsf{f}(X)$, the set of finite subsets of X, where the action is defined point-wise:
$$m \cdot S := \{m \cdot x \mid x \in S\} \qquad \text{for } m \in M, S \in \mathcal{P}_\mathsf{f}X.$$
3. Every set D can be equipped with the *discrete* M-set structure: $m \cdot d = d$ for all $m \in M, d \in D$.
4. The set M itself is an M-set with monoid multiplication (from the left) as the action. When defining the M-set action as a homomorphism (Remark 2.3), this action corresponds to the monoid homomorphism
$$h\colon M \to (M^M, \cdot, \mathsf{id}_M) \qquad h(m) = (\ell \mapsto m \cdot \ell) \in M^M.$$

Our leading examples for the atoms are $A := \mathbb{A}$ where $\mathbb{A}$ is a countably infinite set interpreted as *names* and $A := \mathbb{Q}$, the set of rational numbers.

Example 2.5. The main instances of monoids M for nominal techniques studied in the literature are the following:

1. For a set A, let $\mathfrak{S}_\mathsf{f}(A)$ be the set of bijections $\phi\colon A \to A$ that fix all but finitely many $a \in A$ (i.e. $\phi(a) = a$ for almost all $a \in A$). For the prime example of nominal sets (over the equality symmetry), one considers $M := \mathfrak{S}_\mathsf{f}(\mathbb{A})$ [15, 28].
2. For the set $\mathbb{Q}$ of rational numbers, let $\mathsf{Aut}(\mathbb{Q},<)$ be the set of bijective and monotone maps $\mathbb{Q} \to \mathbb{Q}$. For nominal sets over the total order symmetry, one considers $M := \mathsf{Aut}(\mathbb{Q},<)$ [7].
3. Let $\mathsf{Fin}(A) \subseteq A \to A$ be the set of maps that fix all but finitely many elements, i.e. for every $f \in \mathsf{Fin}(A)$, the set $\{f(a) \ne a \mid a \in A\}$ is finite (hence, $\mathfrak{S}_\mathsf{f}(A)$ is a submonoid of $\mathsf{Fin}(A)$). Then, *nominal renaming sets* [17] are based on M-set for $M := \mathsf{Fin}(\mathbb{A})$.

The action of an M-set X renames the atoms A that are buried in an element $x \in X$. Intuitively, an M-set is *nominal*, if each $x \in X$ carries only finitely many atoms. We use the following notation when defining nominal sets and related notions later:

Definition 2.6. We say that $m, m' \in M$ are *identical on* $S \subseteq A$, written $m \approx_S m'$, if $m(a) = m'(a)$ for all $a \in S$.

Lemma 2.7. *For every $S \subseteq A$, $\approx_S$ is an equivalence relation and a congruence w.r.t. multiplication from the left:*
$$m \approx_S m' \text{ implies } \ell \cdot m \approx_S \ell \cdot m' \qquad \text{for all } m, m', \ell \in M.$$

Intuitively, $\approx_S$ describes that two monoid elements are indistinguishable when restricted to $S \subseteq A$. This can be used to define, which atoms $S \subseteq A$ are carried by the element x of an M-set X:

Definition 2.8 [17, Def. 7]**.** A set $S \subseteq A$ *supports* $x \in X$ of an M-set X, if for all $m \approx_S m'$ (in M), we have $m \cdot x = m' \cdot x$. An M-set X is *nominal* if every element of X is supported by a finite set $S \subseteq A$.

We also say that $x \in X$ has finite support. If M happens to be a group, the definition of support simplifies:

Lemma 2.9. *Given that M is a group, $S \subseteq A$ supports $x \in X$ iff for all $m \in M$ with $\forall a \in S\colon m(a) = a$, we have $m \cdot x = x$.*

The simpler characterization of support in Lemma 2.9 also holds in the case $M = \mathsf{Fin}(A)$ [17, Lem. 13].

Example 2.10. Most of the M-sets from Example 2.4 are nominal:

1. A is a nominal M-set: every $a \in A$ is supported by $S := \{a\}$, and also by any superset of S. If we consider $M := \mathfrak{S}_\mathsf{f}(\mathbb{A})$, $A := \mathbb{A}$, then a is also supported by the infinite set $\mathbb{A} \setminus \{a\}$; hence, finiteness is required in the definition of nominal M-sets.
2. If X is a nominal M-set, then so is $\mathcal{P}_\mathsf{f}(X)$. A set $E \in \mathcal{P}_\mathsf{f}(X)$ of elements is supported by the union of finite supports of the elements $x \in E$. This union is again finite because E is finite.
3. Every discrete M-set D is nominal because every element $x \in D$ is supported by the empty set.
4. For all monoids M of interest for nominal techniques, M considered as an M-set (Example 2.4.4) is not nominal: whenever M is a nominal M-set, then *every* M-set

is nominal. For example, $\mathfrak{S}_f(\mathbb{A})$ is not nominal because no $\sigma \in \mathfrak{S}_f(\mathbb{A})$ has finite support.

Remark 2.11. It is a central property in the theory of nominal sets that every element has a *least* finite set supporting it (least w.r.t. set inclusion). Whether all elements of nominal M-sets have least finite supports is a property that depends on the choice of M. It indeed holds for our main examples for equality [16, 28], total order [7], and renaming [17], but the proofs are fairly different. If all elements in an M-set have a least finite support, it is often sloppily called least supports, leaving finiteness implicit (note that for $M = \mathfrak{S}_f(\mathbb{A})$, every element $a \in \mathbb{A}$ is supported by the infinite set $\mathbb{A} \setminus \{a\}$, so $\{a\}$ is not the least support, but only the least *finite* support).

Definition 2.12 [7, Definition 4.11]**.** A monoid $M \leq A^A$ *admits least supports* if each element of a nominal M-set has a least finite support. If so, we write $\mathsf{supp}_X : X \to \mathcal{P}_f(A)$ for the map that sends elements of the M-set X to their least finite support. We simply write supp if the M-set is clear from the context.

As mentioned, all monoids from Example 2.5 admit least supports. Intuitively, $\mathsf{supp}(x) \subseteq A$ can be understood as precisely the atoms that appear in x. For example, $\mathsf{supp}_A : A \to \mathcal{P}_f(A)$ sends a to $\{a\}$, and $\mathsf{supp}_{\mathcal{P}_f(X)} : \mathcal{P}_f(X) \to \mathcal{P}_f(A)$ is

$$\mathsf{supp}_{\mathcal{P}_f(X)}(E) = \bigcup \{\mathsf{supp}_X(x) \mid x \in E\}.$$

The opposite of an atom $a \in A$ in the support is:

Definition 2.13. An atom $a \in A$ is *fresh* (notated $a \# x$) for an element $x \in X$ in a nominal set X, if $a \notin \mathsf{supp}_X(x)$. We also write $a \# x, y, \ldots$ for multiple elements $x, y, \ldots$, meaning that a is fresh for all those.

Lemma 2.14. *Equivariant functions $f : X \to Y$ preserve supports: if S supports x, then it also supports $f(x)$. If X and Y have least finite supports, then* $\mathsf{supp}_Y(f(x)) \subseteq \mathsf{supp}_X(x)$.

Intuitively, equivariant maps can possibly forget about atoms, but can never introduce new atoms. The support map $\mathsf{supp}_X : X \to \mathcal{P}_f A$ itself is not equivariant in general, but it is if M is a group:

Lemma 2.15 [17, Lem. 11]**.** *If $(X, \cdot)$ has least supports, then we have for all $m \in M$ and $x \in X$:*
1. $\mathsf{supp}_X(m \cdot x) \subseteq m \cdot \mathsf{supp}_X(x)$.
2. *If m is invertible, then* $\mathsf{supp}_X(m \cdot x) = m \cdot \mathsf{supp}_X(x)$.

In nominal sets, the finiteness notion is in terms of orbits:

Definition 2.16. Given a group M, the *orbit* of an element x in an M-set is the subset $\{m \cdot x \mid m \in M\} \subseteq X$. A nominal M-set is *orbit-finite* if it consists of finitely many orbits.

If M is a group, then every M-set X is a disjoint union of orbits, and X is called *oribit-finite* if it consists of finitely many orbits. In nominal automata, one requires the state space to be orbit-finite, as opposed to plain finiteness in classical automata theory. In general, orbit-finite objects are infinite – except for discrete nominal sets.

Definition 2.17. For $M \leq A^A$, let $\mathsf{Nom}(M)$ be the category of nominal M-sets and equivariant maps.

Example 2.18. We fix names of the standard instances:
1. The category of *nominal sets* is denoted by $\mathsf{Nom} = \mathsf{Nom}(\mathfrak{S}_f(\mathbb{A}))$ (slightly overloading notation), that is for $A := \mathbb{A}$ and $M := \mathfrak{S}_f(\mathbb{A})$ modelling equality symmetry.
2. The category of *ordered nominal sets* is $\mathsf{OrdNom} = \mathsf{Nom}(\mathsf{Aut}(\mathbb{Q}, <))$, i.e. for $A := \mathbb{Q}$, $M := \mathsf{Aut}(\mathbb{Q}, <)$.
3. *Nominal renaming sets* are denoted by $\mathsf{RnNom} = \mathsf{Nom}(\mathsf{Fin}(\mathbb{A}))$, i.e. for $A := \mathbb{A}$, $M := \mathsf{Fin}(\mathbb{A})$.

3 Supported Sets

The central notion of the present paper are supported sets, which are parametric in the set A of atoms or data symbols of interest:

Definition 3.1. For a set A, the category of *supported sets* $\mathsf{Supp}(A)$ contains the following:
1. A supported set X consists of a set X together with a map $\mathsf{s}_X : X \to \mathcal{P}_f(A)$.
2. A supported map $f : (X, \mathsf{s}_X) \to (Y, \mathsf{s}_Y)$ is a map $f : X \to Y$ with $\mathsf{s}_Y(f(x)) \subseteq \mathsf{s}_X(x)$ for all $x \in X$. This means, f makes the following triangle weakly commute:

$$\begin{tikzcd} X \arrow[rr, "f"] \arrow[dr, "\mathsf{s}_X"'] & \supseteq & Y \arrow[dl, "\mathsf{s}_Y"] \\ & \mathcal{P}_f(A) & \end{tikzcd}$$

Whenever clear from the context, we simply speak of supported sets $X, Y \in \mathsf{Supp}(A)$ and supported maps $f : X \to Y$, leaving s_X implicit.

The intuition for supported sets comes from the least finite support map supp of nominal sets. Hence, in a supported set X, the map $\mathsf{s}_X(x)$ tells which atoms are carried by $x \in X$, but we can not permute or rename them in general.

The main intuition for the morphisms in supported sets comes from the property of equivariant maps that they possibly forget about atoms in the support, but they can never introduce new atoms (Lemma 2.14). This observation becomes the defining property in the definition of supported maps.

Example 3.2. The set A itself is a support set with $\mathsf{s}_A(a) = \{a\}$. However, the singleton subset $\{a\} \subseteq A$ is also a supported set with $\mathsf{s}_{\{a\}}(a) = \{a\}$.

In general, every nominal M-set X (for M admitting least supports) yields a supported set by putting $\mathsf{s}_X := \mathsf{supp}_X$.

The difference between nominal and supported sets is that supp in a nominal set X is a derived notion since it is implicit in the M-set action. On the other hand, for supported sets,

s_X is part of the syntactical structure. For the sake of clarity, we use different mathematical symbols for the semantical supp and the structural s.

Lemma 3.3. *For every $M \leq A^A$, there is a faithful functor $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ sending $(X, \cdot)$ to a supported set on X and sending equivariant maps to their underlying map. If $M \leq A^A$ admits least supports, then $\mathsf{s}_{UX} = \mathsf{supp}_X$.*

Later, we show that this forgetful functor $\mathsf{Nom}(M) \to \mathsf{Supp}(A)$ is right-adjoint and even monadic, so one can consider nominal sets as algebras on supported sets. Before, we establish some categorical properties of $\mathsf{Supp}(A)$ (generic in the choice of A) that will come in handy.

3.1 Universal Constructions and Finiteness

Unsurprisingly, supported sets have a very set-like nature. In the present Section 3.1, we let U denote the forgetful functor $U\colon \mathsf{Supp}(A) \to \mathsf{Set}$ that forgets the support map and sends (X, s_X) to the plain set X. Conversely, every set can be equipped with a trivial support map:

Lemma 3.4. *The inclusion functor defined by*

$$J\colon \mathsf{Set} \hookrightarrow \mathsf{Supp}(A) \qquad JX = (X, \mathsf{s}_X) \text{ with } \mathsf{s}_X(x) = \emptyset.$$

is right-adjoint to the forgetful $U\colon \mathsf{Supp}(A) \to \mathsf{Set}$ ($U \dashv J$) and $UJ = \mathsf{Id}_{\mathsf{Set}}$.

In other words, Set is a reflective subcategory of $\mathsf{Supp}(A)$. Similarly to sets, universal constructions such as limits and colimits all exist in supported sets and are (almost) set-like. Given a diagram $D\colon \mathcal{D} \to \mathsf{Set}$, we write $\mathsf{pr}_X\colon \lim D \to DX$ for the limit projection map of $X \in \mathcal{D}$ and $\mathsf{in}_X\colon DX \to \mathrm{colim} D$ for the colimit injections.

Proposition 3.5. *All colimits in $\mathsf{Supp}(A)$ exist and are preserved by $U\colon \mathsf{Supp}(A) \to \mathsf{Set}$: Given a diagram $D\colon \mathcal{D} \to \mathsf{Supp}(A)$, the colimit is formed in Set and then equipped with the support $\mathsf{s}\colon \mathrm{colim}\, UD \to \mathcal{P}_\mathsf{f}(A)$:*

$$\mathsf{s}(c) = \bigcap \{\mathsf{s}_{DY}(y) \mid Y \in \mathcal{D}, y \in DY, \mathsf{in}_Y(y) = c\}.$$

The intersection handles the case where elements of possibly different support are identified in the colimit. In this case, only the atoms present in all source elements survive. The intersection vanishes if there are no connecting morphisms in the diagram:

Example 3.6. The coproduct of supported sets X, Y is given by their disjoint union, $X + Y$ equipped with support $\mathsf{s}_{X+Y}(\mathsf{in}_X(x)) = \mathsf{s}_X(x)$ and $\mathsf{s}_{X+Y}(\mathsf{in}_Y(y)) = \mathsf{s}_Y(y)$. Every supported set is the coproduct of singleton supported sets, i.e. it is locally finite, as we prove later.

Limits on the other hand are not formed entirely as in Set, because supported sets always have a finite support by definition:

Proposition 3.7. *$\mathsf{Supp}(A)$ is complete. The limit of a diagram $D\colon \mathcal{D} \to \mathsf{Supp}(A)$ is the subset of finitely supported elements in the limit $\lim UD$ in Set:*

$$\lim D = \{x \in \lim UD \mid \bigcup_{Y \in \mathcal{D}} \mathsf{s}_{DY}(\mathsf{pr}_Y(x)) \text{ is finite}\} \qquad (1)$$

The process of restricting to elements that are finitely supported is precisely what also happens in limits in nominal sets. Consequently, the side-condition disappears for finite limits:

Corollary 3.8. *$U\colon \mathsf{Supp}(A) \to \mathsf{Set}$ preserves all finite limits.*

$\mathsf{Supp}(A)$ is cartesian closed, that is, we have an internal hom object. Similarly to nominal set, this internal hom object X^E in $\mathsf{Supp}(A)$ contains more than just the plain hom set $\hom(E, X)$:

Definition 3.9. For a supported set E, define the functor $(-)^E\colon \mathsf{Supp}(A) \to \mathsf{Supp}(A)$ by

$$X^E := \{f\colon E \to X \mid \bigcup_{e \in E} \mathsf{s}_X(f(e)) \setminus \mathsf{s}_E(e) \text{ is finite}\}$$

$$\mathsf{s}_{X^E}(f) = \bigcup_{e \in E} \mathsf{s}_X(f(e)) \setminus \mathsf{s}_E(e)$$

For supported maps $g\colon X \to Y$ we put

$$g^E\colon X^E \to Y^E \qquad g^E(f) = g \cdot f$$

Remark 3.10. If E is a finite supported set, then X^E contains all maps $E \to X$.

Proposition 3.11. *For every supported set E, the functor $(-) \times E$ is left adjoint to $(-)^E$, i.e. $(-) \times E \dashv (-)^E$.*

In contrast to nominal sets, categorical finiteness notions in $\mathsf{Supp}(A)$ express actual finiteness. For the present paper, we do not need the precise definition of finitely presentable objects and locally finitely presentable (lfp) categories [3, 18]. For the present purposes, it is enough to mention that $\mathsf{Supp}(A)$ is lfp and that the finitely presentable objects are the finite supported sets, i.e. $\mathsf{Supp}(A)$ is locally finite. Detailed definitions can be found in the proof.

Proposition 3.12. *$\mathsf{Supp}(A)$ is an lfp category, where the finitely presentable objects are the finite supported sets.*

3.2 Injectivity, Surjectivity, Regularity

Notions of surjective and injective maps generalize from Set:

Lemma 3.13. *Monomorphisms in $\mathsf{Supp}(A)$ are precisely the injective supported maps and epimorphisms the surjective supported maps.*

However, not all bijective supported maps are isomorphisms in $\mathsf{Supp}(A)$, because they may lack the following property:

Definition 3.14. A map $f\colon X \to Y$ is called *support-reflecting* if $\mathsf{s}_Y \cdot f = \mathsf{s}_X$. Intuitively, support-reflecting means that f is a supported map and does not drop any atoms.

Example 3.15. Assume that the set of atoms A is countably infinite. Then we have a bijective map $b\colon A \to \mathbb{N}$. With supports $\mathsf{s}_A(a) = \{a\}$ and $\mathsf{s}_{\mathbb{N}}(n) = \emptyset$, the bijection b is a supported map, but not an isomorphism in $\mathsf{Supp}(A)$.

Proposition 3.16. *The isomorphisms in* $\mathsf{Supp}(A)$ *are precisely the support-reflecting bijective maps.*

The notion of support-reflection also affects the image factorization of supported maps: in sets, every map $f\colon X \to Y$ has a unique image factorization

$$\begin{tikzcd} X \arrow[r, two heads, "e"] \arrow[rr, bend left, "f"] & \mathsf{Im}(f) \arrow[r, tail, "m"] & Y \end{tikzcd}$$

into a surjective map e and an injective map m through a set $\mathsf{Im}(f)$, the image of f. Considering a supported map $f\colon X \to Y$, there are multiple options how to define the support map $\mathsf{s}_{\mathsf{Im}(f)}$ on the image of f. There are two supported sets from which $\mathsf{Im}(f)$ can inherit its structure from:

1. From X: $\mathsf{s}_{\mathsf{Im}(f)}(i) = \bigcap \{\mathsf{s}_X(x) \mid e(x) = i\}$.
2. From Y: $\mathsf{s}_{\mathsf{Im}(f)}(i) = \mathsf{s}_Y(m(i))$.

In the first option, e is a regular epimorphism, and in the second option m is a regular monomorphism. In Set and Nom (and in fact any topos) all epimorphisms and all monomorphisms are regular, but in supported sets they differ. In supported sets, we can characterize regular epimorphisms and monomorphisms as follows (see [2, Rem 7.76(2)] for an overview of monomorphism and epimorphism notions).

Proposition 3.17. *An epimorphism* $e\colon X \to Y$ *is regular iff* $\mathsf{s}_Y(y) = \bigcap\{\mathsf{s}_X(x) \mid e(x) = y\}$.

Proposition 3.18. *A monomorphism is regular iff it is support-reflecting.*

The supported map $t\colon 1 \to 2$, $0 \mapsto 1$ *is a regular subobject classifier. This means that for every regular monomorphism* $m\colon S \to X$, *there is a unique supported map* $\chi_S\colon X \to 2$ *such that*

$$\begin{tikzcd} S \arrow[r, "m"] \arrow[d, "!"'] & X \arrow[d, "\chi_S"] \\ 1 \arrow[r, "t"] & 2 \end{tikzcd}$$

is a pullback square.

Note that this is a weaker notion than a subobject classifier and thus weaker than what we have in Nom and Set. Thus, $\mathsf{Supp}(A)$ is not a topos.

In summary, for every set A, $\mathsf{Supp}(A)$ is a complete, cocomplete, cartesian closed, and locally finite category, in which epis resp. monos are injective resp. surjective maps; and regular monomorphisms have subobject classifiers: thus, we can make use of this categorical structure and characterizations, when considering algebraic theories on them, such as nominal sets.

4 Monadicity of Nominal M-Sets

If a category is *monadic*, i.e. if it is intuitively a well-behaved class of algebras, it becomes applicable for many results and constructions, e.g. regarding the representation and the generalized powerset construction. The category of M-sets is monadic over Set, but $\mathsf{Nom}(M)$ fails to be monadic over Set in the instances of interest (Example 2.18). In the present section, we show that it is indeed monadic, but *over supported sets.*

Algebraic theories have been studied extensively throughout the decades, and there are several equivalent ways how to define an algebraic theory. One of them is via monads: given a monad T on a category C, e.g. Set, its *Eilenberg-Moore category* $\mathsf{EM}(T)$ contains the models of the algebraic theory defined by T (see [4, 10.3] for an introduction). The forgetful functor $U^T\colon \mathsf{EM}(T) \to C$ is a right-adjoint functor and its left adjoint $F^T\colon C \to \mathsf{EM}(T)$ sends an object $X \in C$ (e.g. a set of generators) to the *free algebra* on X; for $C = \mathsf{Supp}(A)$:

$$F^T\colon \mathsf{Supp}(A) \to \mathsf{EM}(T) \quad\dashv\quad U^T\colon \mathsf{EM}(T) \to \mathsf{Supp}(A)$$

For example, since infinite products in $\mathsf{Nom}(M)$ are different than in set, the forgetful functor $\mathsf{Nom}(M) \to \mathsf{Set}$ is not right-adjoint and therefore not monadic:

Definition 4.1. A functor $U\colon \mathcal{D} \to C$ is called *monadic (over C)* if there is a monad T on C such that $\mathcal{D}$ is $\mathsf{EM}(T)$ and U is the forgetful functor.

As we show, the functor to supported sets $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ is right-adjoint and even monadic. So we can view nominal sets as algebras over $\mathsf{Supp}(A)$. There, the algebraic operations come from the following (supported) set:

Definition 4.2. For $M \le A^A$ and a set $S \subseteq A$, put $[m]_S$ for the $\approx_S$-equivalence class of m (cf. Definition 2.6) and

$$M/S = \{[m]_S \mid m \in M\}$$

for the set of equivalence classes. Moreover, M/S is a supported set by putting $\mathsf{s}_{M/S}([m]_S) := m[S]$.

Remark 4.3. Note that M/S is the image of the composition $M \rightarrowtail A^A \twoheadrightarrow A^S$ of the submonoid inclusion $M \rightarrowtail A^A$ and the restriction of maps $A^A \twoheadrightarrow A^S$ to $S \subseteq A$:

$$\begin{tikzcd} M \arrow[r, tail] \arrow[d, two heads] & A^A \arrow[d, two heads] \\ M/S \arrow[r, tail] & A^S \end{tikzcd} \quad \text{(in sets)} \qquad (2)$$

Hence, one can consider M/S either as equivalence classes of M-elements that are identical on S or as special maps $S \to A$ that are obtained by restricting some $m \in M$ to $S \subseteq A$.

Lemma 4.4. *M/S with the action $\ell \cdot [m]_S := [\ell \cdot m]_S$ is a nominal M-set in which $m[S]$ supports $[m]_S$.*

Intuitively, M/S is the free nominal M-set on one generator with support S. Note that Lemma 4.4 does not make any statement about the existence of a *least* finite support. For this, we use the following condition on M:

Definition 4.5. A monoid $M \leq A^A$ is called *lock-free* if for all finite $R \subseteq A$ and $a \in A \setminus R$, there is some $\ell \in M$ with $\ell \approx_R \mathsf{id}_A$ and $\ell(a) \neq a$.

Intuitively, the condition says that whenever we have an element with support $R \cup \{a\}$, then there is some $\ell \in M$ that replaces a by something fresh while fixing the rest of the support.

Example 4.6. All the leading examples (Ex. 2.5) have lock-free monoids.

- For $M := \mathfrak{S}_{\mathsf{f}}(\mathbb{A})$ and $M := \mathsf{Fin}(\mathbb{A})$, consider a finite $R \subseteq \mathbb{A}$ and $a \mathrel{\#} R$. Then for $b \mathrel{\#} a, R$, the permutation $\ell = (a\,b)$ fulfils the desired property $(a\,b) \approx_R \mathsf{id}_{\mathbb{A}}$.
- For $M := \mathsf{Aut}(\mathbb{Q}, <)$, the verification uses a notion called homogeneity [35, Lemma 5.2].

When using the lock-free monoids in proofs, we will use its contrapositive:

Lemma 4.7. *$M \leq A^A$ is lock-free iff for all $a \in A$ and finite $R \subseteq A$ we have that*

$$(\forall \ell \in M, \ell \approx_R \mathsf{id}_A \colon \ell(a) = a) \quad \textit{implies} \quad a \in R. \tag{3}$$

Lemma 4.8. *If M is lock-free, then $\mathsf{supp}_{M/S}([m]_S) = m[S]$.*

We can now define what the monad does on a given supported set X:

Definition 4.9. The *free nominal M-set* is the functor $M\bullet\colon \mathsf{Supp}(A) \to \mathsf{Supp}(A)$ sending X to the supported set

$$M\bullet X = \coprod_{x \in X} M/\mathsf{s}(x) = \{([m]_{\mathsf{s}(x)}, x) \mid x \in X, m \in M\}$$

and a supported map $f\colon X \to Y$ to the supported map

$$\begin{aligned} &M\bullet f\colon M\bullet X \to M\bullet Y \\ &(M\bullet f)([m]_{\mathsf{s}_X(x)}, x) = ([m]_{\mathsf{s}_Y(f(x))}, f(x)). \end{aligned}$$

The notation $M\bullet$ stems from the intuition of freely extending X with an M-set action. Every coproduct component $M/\mathsf{s}(x)$ has the support map as defined in Definition 4.2, and $M\bullet X$ is the coproduct of such supported sets:

$$\mathsf{s}_{M\bullet X}([m]_{\mathsf{s}(x)}, x) = m[\mathsf{s}(x)].$$

Every coproduct component is a nominal M-set and so is the whole $M\bullet X$:

Proposition 4.10. *$M\bullet X$ with the M-set action*

$$\ell \cdot ([m]_{\mathsf{s}(x)}, x) = ([\ell \cdot m]_{\mathsf{s}(x)}, x).$$

is nominal and gives rise to a functor

$$F\colon \mathsf{Supp}(A) \to \mathsf{Nom}(M) \qquad FX = M\bullet X.$$

For a supported set X, every element $([m]_{\mathsf{s}(x)}, x)$ of the nominal set $M\bullet X$ is equivalently an element $x \in X$ together with an $|\mathsf{s}(x)|$-tuple of atoms. Of course, such an equivalence

$$\begin{tikzcd}[column sep=large]
\overbrace{\mathsf{Nom}(\mathsf{Aut}(\mathbb{Q},<))}^{\mathsf{OrdNom}} \arrow[dr, bend left, "U"] \arrow[dr, phantom, "\dashv"] & \overbrace{\mathsf{Nom}(\mathfrak{S}_{\mathsf{f}}(\mathbb{A}))}^{\mathsf{Nom}} \arrow[d, bend left, "U"] \arrow[d, phantom, "\dashv"] & \overbrace{\mathsf{Nom}(\mathsf{Fin}(\mathbb{A}))}^{\mathsf{RnNom}} \arrow[dl, bend left, "U"] \arrow[dl, phantom, "\dashv"] \\
& \boxed{\mathsf{Supp}(A)} \arrow[ul, bend left, "\mathsf{Aut}(\mathbb{Q},<)\bullet(-)"] \arrow[u, bend left, "\mathfrak{S}_{\mathsf{f}}(\mathbb{A})\bullet(-)"] \arrow[ur, bend left, "\mathsf{Fin}(\mathbb{A})\bullet(-)"] &
\end{tikzcd}$$

Figure 1. Monadic adjunctions between leading examples of nominal M-sets and $\mathsf{Supp}(A)$ for countably infinite A.

class $[m]_{\mathsf{s}(x)}$ is not an arbitrary tuple of elements of A but only one that is obtained from restricting some $m \in M \subseteq (A \to A)$ to $\mathsf{s}_X(x) \subseteq A$ (cf. (2)). For example, for the equality symmetry $M := \mathfrak{S}_{\mathsf{f}}(\mathbb{A})$, such a $t \in M/\mathsf{s}_X(x)$ is a tuple of distinct elements of A.

Proposition 4.11. *If $M \leq A^A$ admits least supports, then $F\colon \mathsf{Supp}(A) \to \mathsf{Nom}(M)$ is left-adjoint to $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ with unit $\eta_X\colon X \to UFX$, $\eta_X(x) = ([\mathsf{id}_A]_{\mathsf{s}(x)}, x)$.*

This adjunction shows that every nominal M-set is an Eilenberg-Moore algebra and that every supported set generates a free nominal M-set with the following universal property:

Corollary 4.12. *Given a supported set X and a nominal M-set (where M admits least supports), every map*

$$f\colon X \to Y \quad \textit{with} \quad \mathsf{supp}_Y(f(x)) \subseteq \mathsf{s}_X(x) \quad \forall x \in X$$

uniquely extends to an equivariant map

$$g\colon M\bullet X \to Y \quad \textit{with} \quad g([\mathsf{id}]_{\mathsf{s}(x)}, x) = f(x) \quad \forall x \in X.$$

We use Beck's monadicity theorem (see e.g. [25, Section VI.7]) to prove the monadicity of the adjunction between $\mathsf{Nom}(M)$ and $\mathsf{Supp}(A)$:

Theorem 4.13. *If M admits least supports and is lock-free, $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ is monadic.*

Concretely, this proves the monadicity of our leading examples:

Example 4.14. The following categories are monadic over $\mathsf{Supp}(A)$, for A being countably infinite (visualized in Fig. 1):

1. The category Nom of nominal sets (with equality symmetry) is monadic over $\mathsf{Supp}(\mathbb{A})$. The operations on a generator x in the corresponding theory are injective maps $\mathsf{s}(x) \to \mathbb{A}$, i.e. the monad instantiates to:
$$\mathfrak{S}_{\mathsf{f}}(\mathbb{A})\bullet X = \big\{(\pi, x) \mid x \in X, \pi\colon \mathsf{s}_X(x) \rightarrowtail \mathbb{A}\big\}$$
2. The category RnNom of nominal renaming sets is monadic over $\mathsf{Supp}(\mathbb{A})$. The operations on a generator x are arbitrary maps $\mathsf{s}(x) \to \mathbb{A}$ (not necessarily injective), i.e. the monad instantiates to:
$$\mathsf{Fin}(\mathbb{A})\bullet X = \big\{(\pi, x) \mid x \in X, \pi\colon \mathsf{s}_X(x) \to \mathbb{A}\big\}$$

3. The category OrdNom of nominal sets for the total order symmetry is monadic over $\mathsf{Supp}(\mathbb{Q})$ (note that $\mathbb{Q} \cong A$ for A being countably infinite). The operations on a generator x are injective maps $\mathsf{s}(x) \to \mathbb{Q}$ preserving the order. The monad instantiates to:
$$\begin{aligned}\mathsf{Aut}(\mathbb{Q}, <)\bullet X = \big\{(\pi, x) \mid x \in X, \pi\colon \mathsf{s}_X(x) \to \mathbb{Q}\\ \forall q,p \in \mathsf{s}_X(x) :\\ q < p \Rightarrow \pi(q) < \pi(p)\big\}\end{aligned}$$

As a direct application of the monadicity, we can characterize orbit-finite nominal sets, resp. finitely presentable objects in $\mathsf{Nom}(M)$ using a general result about algebraic categories [1, Thm. 3.7]:

Example 4.15. A nominal set X is orbit-finite iff it can be described by a finite supported set G of *generators* and a finite subset
$$E \subseteq (M\bullet G) \times (M\bullet G)$$
of *equations*. Given such finite G and E, the projection maps
$$\begin{tikzcd} E \arrow[r, shift left, "\ell"] \arrow[r, shift right, "r"'] & M\bullet G \end{tikzcd} \qquad \text{in } \mathsf{Supp}(A)$$
uniquely extend to equivariant maps
$$\begin{tikzcd} M\bullet E \arrow[r, shift left, "\bar\ell"] \arrow[r, shift right, "\bar r"'] & M\bullet G \end{tikzcd} \qquad \text{in } \mathsf{Nom}(M)$$
using the adjunction of the monadic $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$. Then, their coequalizer is the orbit-finite X. The characterization says that 1. every orbit-finite nominal set can be described by such finite supported sets G and E, and 2. that any such finite system of equations E on G yields an orbit-finite nominal set.

For example, for $M := \mathfrak{S}_\mathsf{f}(\mathbb{A})$, the nominal set of unordered pairs of atoms can be described by one generator g
$$G = \{g\} \qquad \mathsf{s}_G(g) := \{a, b\} \qquad \text{for fixed } a, b \in \mathbb{A}$$
and one equation
$$E := \big\{(a\,b)\cdot g = \mathsf{id}\cdot g\big\}.$$
Here, we use intuitive notation to denote
$$E := \big\{\big(([(a\,b)]_{\{a,b\}}, g),\ ([\mathsf{id}]_{\{a,b\}}, g)\big)\big\} \subseteq (M\bullet G)\times(M\bullet G).$$
Let $\ell, r\colon E \to M\bullet G$ denote the projections (in $\mathsf{Supp}(A)$) and $\bar\ell, \bar r$ denote their extensions to Nom; then
$$\begin{tikzcd} M\bullet E \arrow[r, shift left, "\bar\ell"] \arrow[r, shift right, "\bar r"'] & M\bullet G \arrow[r, two heads] & \big\{\{a,b\} \mid a, b \in \mathbb{A}, a \neq b\big\} \end{tikzcd}$$
is a coequalizer in Nom.

5 Abstraction and de Bruijn indices

In λ-calculus, the computational steps (β-reduction) essentially consist of a substitution rule
$$(\lambda x.\, T)\, P \longrightarrow_\beta T[x := P]$$
which causes trouble if some bound variable names in T are not sufficiently fresh. Thus, it might be necessary to rename those bound variables in T (called α-conversion) before a β step can be performed:
$$\lambda y.\, S \longrightarrow_\alpha \lambda z.\, S[y := z]$$
Here, the same side-condition on z and the bound variables in S needs to be taken care of.

This requirement of freshness is a troublesome side condition, because it makes the substitution a partial function, or put differently, because β-reduction would lead to wrong results if the freshness condition was neglected.

Thus, there are several approaches to define λ-expressions directly modulo α-equivalence, making substitution total and β-reduction always applicable to λ-expressions containing a reducible expression (redex), i.e. expressions of the form:
$$(\lambda x.\, T)\, P$$
In 1972, de Bruijn [10] invented a technique of replacing the variable names with an index that specifies to which binder it belongs. The *de Bruijn index* of a variable x counts the number binders between the variable and its corresponding binder λx, when considering the λ-expression as a tree.

In 1999, Gabbay and Pitts [15] presented another way to define λ-expressions directly as α-equivalence classes, using *nominal sets* (called *FM-sets* back then). In their LICS paper, they define λ-abstraction as a functor on the category of nominal sets.

In the present chapter, we define an abstraction functor on supported sets
$$\mathcal{B}\colon \mathsf{Supp}(A) \to \mathsf{Supp}(A)$$
that internally uses de Bruijn indices and that requires A to be a countably infinite set. This functor in fact has a lifting to nominal sets and the lifting is (naturally isomorphic to) the abstraction functor $[\mathbb{A}]\colon \mathsf{Nom} \to \mathsf{Nom}$ by Gabbay and Pitts [15].

Assumption 5.1. For the rest of the paper, fix $A := \mathbb{A}$ and $M := \mathfrak{S}_\mathsf{f}(\mathbb{A})$. Hence, we may assume a bijection $\varrho\colon \mathbb{N} \to \mathbb{A}$ and we simply write Nom for the M-nominal sets.

First, let us recall the definition of the abstraction functor on nominal sets (see [15, 28] for details).

Definition 5.2 [15]. For a nominal set X, the equivalence relation $\sim_\alpha$ on $\mathbb{A}\times X$ is defined by
$$(a, x) \sim_\alpha (b, y) \quad :\Leftrightarrow \quad \exists c \,\#\, (a, b, x, y)\colon (c\,a)\cdot x = (c\,b)\cdot y$$
The *abstraction functor* $[\mathbb{A}]X\colon \mathsf{Nom} \to \mathsf{Nom}$ is given by
$$[\mathbb{A}]X = (A \times X)/\sim_\alpha$$
We write $\langle a\rangle x$ for the equivalence class of $(a, x) \in \mathbb{A}\times X$. The definition of $[\mathbb{A}]X$ on morphisms and that of the nominal structure are point-wise: $\pi\cdot\langle a\rangle x = \langle\pi(a)\rangle(\pi\cdot x)$ (which is the equivalence class of $\pi\cdot(a, x)$). As a result of quotienting by α-equivalence, a disappears from the support of $\langle a\rangle x$:
$$\mathsf{supp}_{[\mathbb{A}]X}(\langle a\rangle x) = \mathsf{supp}_X(x) \setminus \{a\}.$$

Example 5.3 [15]**.** The initial algebra of the functor

$$\Lambda X = \mathbb{A} + [\mathbb{A}]X + X \times X$$

is carried by the nominal set of λ-expressions modulo α-equivalence. The first summand $\mathbb{A}$ describes variables $a \in \mathbb{A}$, the second summand $[\mathbb{A}]X$ describes λ-abstractions $\lambda x.T$, where $x \in \mathbb{A}$ and T is a λ-expression, and the third summand $X \times X$ describes the application $T\,S$ of one λ-expression to one other. The inductive definition can be extended to a recursion principle which then allows to define substitution as a total function on λ-expressions modulo α-equivalence.

The definition of $[\mathbb{A}]$ makes use of the $\mathfrak{S}_\mathrm{f}(\mathbb{A})$-action to capture α-equivalence. When defining abstraction as a functor on supported sets, we do not have renaming available, and so we use de Bruijn indices, when introducing abstraction. Here, we make heavy use of the bijection $\varrho\colon \mathbb{N} \to \mathbb{A}$ from Assumption 5.1.

Definition 5.4. The functor $\mathcal{B}\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$ sends a supported set X to the same set $\mathcal{B}X = X$ but with a different support function. To distinguish elements of X and $\mathcal{B}X$, we write $\lambda.x \in \mathcal{B}X$ for $x \in X$ (i.e. x is under the nameless binder λ). The support on $\mathcal{B}X$ is defined by the map

$$\begin{aligned} &\mathsf{s}_{\mathcal{B}X}\colon \mathcal{B}X \to \mathcal{P}_\mathrm{f}(\mathbb{A}) \\ &\mathsf{s}_{\mathcal{B}X}(\lambda.x) := \{\varrho(k) \mid \varrho(k+1) \in \mathsf{s}_X(x), k \in \mathbb{N}\} \qquad (4) \end{aligned}$$

A supported map $f\colon X \to Y$ is sent to the same underlying map

$$\mathcal{B}f\colon \mathcal{B}X \to \mathcal{B}Y \qquad \mathcal{B}f(\lambda.x) = \lambda.f(y).$$

We call $\mathcal{B}$ the *de Bruijn functor.*

The definition of $\mathsf{s}_{\mathcal{B}X}$ captures the idea of de Bruijn indices: We can think of the notation $\lambda.x$ as a lambda abstraction of nameless binder $\lambda.$ of a lambda term x.

- The variables referring to $\lambda.$ have the de Bruijn index of 0 (at the level of x), because there is no other binder between x and '$\lambda.$'. Since $\varrho(0)$ is bound, $\mathsf{s}_{\mathcal{B}X}(x)$ does not depend on whether $\varrho(0) \in \mathsf{s}_X(x)$.
- All other variables $\varrho(k+1) \in \mathsf{s}_X(x)$ refer to variables that are free in $\lambda.x$ and so refer to binders 'more above' than $\lambda.x$. A variable $\varrho(k) \in \mathsf{s}_{\mathcal{B}X}(\lambda.x)$ refers to the same binder as the variable $\varrho(k+1) \in \mathsf{s}_X(x)$ under '$\lambda.$', because the latter is one level further down.

Lemma 5.5. $\mathcal{B}\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$ *is a functor.*

We use a slightly generalized notion of a *lifting* of the functor $\mathcal{B}$ to the category of nominal sets:

Definition 5.6. Given a monad T on a category C and a functor $H\colon C \to C$, a functor $G\colon \mathsf{EM}(T) \to \mathsf{EM}(T)$ is called a *lifting of F* if HU and UG are naturally isomorphic functors:

$$\begin{tikzcd} \mathsf{EM}(T) \arrow[r, "G"] \arrow[d, "U"'] & \mathsf{EM}(T) \arrow[d, "U"] \\ C \arrow[r, "H"] & C \end{tikzcd} \qquad \phi\colon HU \xrightarrow{\cong} UG$$

We say that a lifting is *strict* if ϕ is the identity.

Remark 5.7. Usually, it is required that ϕ is the identity and not just a natural isomorphism. Liftings $G\colon \mathsf{EM}(T) \to \mathsf{EM}(T)$ with equality $UG = HU$ are in one-to-one correspondence to functor over monad distributive laws $TH \to HT$ (i.e. natural transformations that are compatible with the monad structure of T) [20].

In the next lemma we see that this generalization to natural isomorphisms is sound: lifted functors are always naturally isomorphic to a strict lifting:

Lemma 5.8. *Given a natural isomorphism $\phi\colon HU \to UG$ (as in Definition 5.6), then there is a unique strict lifting*

$$\bar{H}\colon \mathsf{EM}(T) \to \mathsf{EM}(T) \qquad (U\bar{H} = HU)$$

such that $\phi\colon H \to G$ is a natural isomorphism in $\mathsf{EM}(T)$*.*

This means that for every $(C, \alpha) \in \mathsf{EM}(T)$, $\phi_{(C,\alpha)}$ is a T-algebra isomorphism $\bar{H}(C,\alpha) \to G(C,\alpha)$.

Theorem 5.9. *The abstraction functor* $[\mathbb{A}]\colon \mathsf{Nom} \to \mathsf{Nom}$ *is a lifting of the de Bruijn functor* $\mathcal{B}\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$*. The natural isomorphism* $\phi\colon \mathcal{B}U \longrightarrow U[\mathbb{A}]$

$$\begin{tikzcd} \mathsf{Nom} \arrow[r, "{[\mathbb{A}]}"] \arrow[d, "U"'] & \mathsf{Nom} \arrow[d, "U"] \\ \mathsf{Supp}(\mathbb{A}) \arrow[r, "\mathcal{B}"] & \mathsf{Supp}(\mathbb{A}) \end{tikzcd}$$

is given by

$$\begin{aligned} &\phi_X\colon \mathcal{B}UX \longrightarrow U[\mathbb{A}]X \\ &\phi_X(\lambda.x) = \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle \varrho(0) \rangle x \end{aligned}$$

where $\mathrm{maxidx}(x) \in \mathbb{N}$ *and* $\sigma_m \in \mathfrak{S}_\mathrm{f}(\mathbb{A})$ *($m \in \mathbb{N}$) are given by:*

$$\mathrm{maxidx}(x) = 1 + \max\{n \in \mathbb{N} \mid \varrho(n) \in \mathsf{s}(x)\}$$

$$\sigma_m(\varrho(\ell)) = \big(\varrho(0) \cdots \varrho(m)\big) = \begin{cases} \varrho(\ell+1) & \text{if } \ell < m \\ \varrho(0) & \text{if } \ell = m \\ \varrho(\ell) & \text{else.} \end{cases}$$

For sufficiently large $m \in \mathbb{N}$, the cycle σ_m behaves identical to the infinite injective map

$$\varrho(\ell) \quad \mapsto \quad \varrho(\ell+1),$$

In the definition of ϕ_X, sufficiently large means $m := \mathrm{maxidx}(x)$, that is, for every $\varrho(k) \in \mathrm{supp}(x)$, we have $k < m$. Similarly, σ_m^{-1} does the shift of indices in the opposite direction (note that $\varrho(0) \notin \mathrm{supp}(\langle \varrho(0) \rangle x)$ in the definition of ϕ_X).

Intuitively, ϕ translates between the de Bruijn indices and the nominal abstraction functor: in the term $\lambda.x \in \mathcal{B}X$, we have $\varrho(0)$ implicitly bound, like it is in $\langle\varrho(0)\rangle(x) \in [\mathbb{A}]X$. However, every $\varrho(k{+}1)$ in x appears as $\varrho(k)$ in the support of $\lambda.x$, so we apply σ_m^{-1} to $\langle\varrho(0)\rangle x$ for sufficiently large $m \in \mathbb{N}$.

With this functor for name binding in supported sets that lifts to nominal sets, we can now apply it to nominal automata.

6 Generalized Determinization

The well-known powerset construction for automata is an instance of a more general principle that can be applied whenever state based systems are extended with side effects. Here, state-based systems are coalgebras for a functor H and side effects are modelled by a monad T. We can apply this principle to take register automata in supported sets and transform them into nominal automata.

The generalized determinization [6, 30], also called generalized powerset construction, instantiates back to the original example of classical automata (with an input alphabet Σ) when considering the Set-functors

$$HX = 2 \times X^\Sigma \qquad \text{and} \qquad TX = \mathcal{P}_\mathsf{f}X.$$

A *coalgebra* (for $H\colon \mathcal{C} \to \mathcal{C}$) is an object $Q \in \mathcal{C}$ together with a morphism $Q \to HQ$. For the above example of H, a deterministic automaton is an H-coalgebra, i.e. a set Q together with a map

$$d\colon Q \to HQ \qquad \text{i.e. } d\colon Q \to 2 \times Q^\Sigma.$$

For a state $q \in Q$, the first component $\mathrm{pr}_1(d(q)) \in 2$ tells whether q is final, and the second component $\mathrm{pr}_2(d(q)) \in Q^\Sigma$ is a map $\Sigma \to Q$ returning the next state on an input symbol $a \in \Sigma$ (the initial state $q_0 \in Q$ is not important for the present work).

On the other hand, a non-deterministic automaton is simply a coalgebra for the composed functor $H\mathcal{P}_\mathsf{f}\colon \mathsf{Set} \to \mathsf{Set}$

$$c\colon Q \to H\mathcal{P}_\mathsf{f}Q \qquad \text{i.e. } c\colon Q \to 2 \times (\mathcal{P}_\mathsf{f}Q)^\Sigma.$$

For each state $q \in Q$ and input symbol $a \in \Sigma$, we obtain a finite set of successor states $\mathrm{pr}_2(c(q))(a) \in \mathcal{P}_\mathsf{f}(Q)$.

The generalized determinization assumes that $T\colon \mathcal{C} \to \mathcal{C}$ is a monad (with unit η and multiplication μ) and that $H\colon \mathcal{C} \to \mathcal{C}$ is a functor that lifts to the Eilenberg-Moore category of T, i.e. that we have a functor $G\colon \mathsf{EM}(T) \to \mathsf{EM}(T)$ with $\phi\colon HU \xrightarrow{\cong} UG$ (Definition 5.6) – for example $HX = 2 \times X^\Sigma$ and $TX = \mathcal{P}_\mathsf{f}X$. Then the generalized determinization turns any HT-coalgebra on Q into a G-coalgebra on TQ, using the adjunction $F \dashv U$ between $\mathcal{C}$ and $\mathsf{EM}(T)$:

$\forall\, c\colon Q \to HTQ$
$\exists!\, d\colon FQ \to GFQ$ in $\mathsf{EM}(T)$
such that:

$$\begin{tikzcd} UFQ \arrow[r, "Ud"] & UGFQ \\ Q \arrow[u, "\eta_Q"] \arrow[r, "c"] & HTQ \arrow[u, "\phi_Q"', "\cong"] \end{tikzcd}$$

Here, $F\colon \mathcal{C} \to \mathsf{EM}(T)$ is the left-adjoint to U. So $T = UF$ and F sends Q to the free T-algebra on Q:

$$FQ = (TQ, \mu_Q)$$

In the case of the powerset construction, $TX = \mathcal{P}_\mathsf{f}X$ is the finite powerset monad with union as multiplication, hence $\mathsf{EM}(T)$ is the category of join-semilattices. The lifting of $HX = 2 \times X^\Sigma$ to $\mathsf{EM}(\mathcal{P}_\mathsf{f})$ is simply given point-wise: for $S \in \mathcal{P}_\mathsf{f}(2 \times X^\Sigma)$ and X a join-semilattice, $\cup S \in HX$ is defined by

$$\cup S := (f', \delta') \quad \text{where} \quad f' := \max\{f \mid (f, \_) \in S\} \ \ (\in 2)$$
$$\delta'(a) := \cup\{\delta(a) \mid (\_, \delta) \in S\} \ \ (\in X).$$

Here, ϕ_Q is the identity. Then, for every non-deterministic automaton $c\colon Q \to 2 \times (\mathcal{P}_\mathsf{f}Q)^\Sigma$, we obtain a deterministic automaton $d\colon \mathcal{P}_\mathsf{f}Q \to 2 \times (\mathcal{P}_\mathsf{f}Q)^\Sigma$ with the following properties (cf. [30] for more details):

1. The state space $\mathcal{P}_\mathsf{f}Q$ of the constructed automaton is the configuration space of the original non-deterministic automaton.
2. The commutative square means that on singleton sets, d is identical to c:
$$c(q) = d(\{q\}) \qquad \text{for all } q \in Q.$$
3. The determinized d is a $\mathcal{P}_\mathsf{f}$-algebra homorphism, i.e. it preserves joins.

Instantiating the generalized determinization to supported and nominal sets ($TX = M\bullet X$) yields:

Construction 6.1. *Given functors $H\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$ and $G\colon \mathsf{Nom} \to \mathsf{Nom}$ with $HU \cong UG$, every coalgebra*

$$c\colon Q \to H(M\bullet Q) \qquad \textit{in } \mathsf{Supp}(\mathbb{A})$$

induces a unique G-coalgebra:

$$d\colon M\bullet Q \to G(M\bullet Q)$$

If Q is finite, then $TQ = M\bullet Q$ is orbit-finite.

The above mentioned properties of the determinization translate into the following:

1. The nominal set $M\bullet Q$ is the configuration space of the original automaton (Q, c).
2. The commutative square means that the configuration $\mathrm{id}_A \cdot q$ in $(M\bullet Q, d)$ behaves like q in (Q, c).
3. The resulting coalgebra $(M\bullet Q, d)$ lives in $\mathsf{Nom}(M)$, i.e. d is an equivariant map.

For the application of this determinization method, we first need functors on Nom that lift from supported sets. We have already seen that the abstraction functor $[\mathbb{A}]$ is such a lifting. Another such functor is the following:

Definition 6.2. The *uniformly supported powerset functor* $\mathcal{P}_\mathsf{ufs}\colon \mathsf{Nom} \to \mathsf{Nom}$ is given by

$$\mathcal{P}_\mathsf{ufs}(X) = \{S \subseteq X \mid \bigcup_{x\in S} \mathrm{supp}_X(x) \text{ is finite}\}$$

On equivariant maps f, $\mathcal{P}_{\mathsf{ufs}}f$ performs the direct image.

Proposition 6.3. *The functors on* Nom *that are the lifting of a functor on* $\mathsf{Supp}(\mathbb{A})$ *contain* $\mathcal{P}_{\mathsf{f}}$, $\mathcal{P}_{\mathsf{ufs}}$, $[\mathbb{A}]$, *and all constant functors and are closed under all (possibly infinite) products and coproducts.*

Example 6.4. Thus, all Nom-functors that arise from the binding signatures [12, 23] are liftings of $\mathsf{Supp}(\mathbb{A})$ functors, in particular, the functor

$$\Lambda\colon \mathsf{Nom} \to \mathsf{Nom} \qquad \Lambda X = \mathbb{A} + [\mathbb{A}]X + X \times X$$

is the lifting of the functor

$$H\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A}) \qquad HX = \mathbb{A} + \mathcal{B}X + X \times X.$$

The coalgebras of Λ are possibly infinite λ-trees modulo α-equivalence. Such a λ-tree can then be represented by a supported set X and a supported map

$$f\colon X \to \mathbb{A} \;+\; \mathcal{B}(M\bullet X) \;+\; (M\bullet X) \times (M\bullet X)$$

Example 6.5. The nominal automata with name binding considered by Schröder et al. [29] are coalgebras for the functor

$$K\colon \mathsf{Nom} \to \mathsf{Nom} \quad KX = 2 \times \mathcal{P}_{\mathsf{ufs}}([\mathbb{A}]X + \mathbb{A} \times X)$$

which is a lifting of $HX = 2 \times \mathcal{P}_{\mathsf{ufs}}(\mathcal{B}X + \mathbb{A} \times X)$ on supported sets. Thus, these nominal automata can be represented by finite HT-coalgebras in $\mathsf{Supp}(\mathbb{A})$.

As a next example, we show that register automata in the style of Cassel et al. [8] straightforwardly adapt to HT-coalgebras in supported sets, where we interpret $\mathbb{A}$ as the names of registers.

Example 6.6. A *(relational) signature* is a set $\mathcal{R}$ with an arity map $\mathsf{ar}\colon \mathcal{R} \to \mathbb{N}$; The supported set of *guards* $\mathcal{G}$ over this relational signature $\mathcal{R}$ is

$$\mathcal{G} = \Big(2 \times \coprod_{r \in \mathcal{R}} \mathbb{A}^{\mathsf{ar}(r)}\Big)^*$$

Intuitively, the words $(-)^*$ represent conjunction, 2 encodes a possible negation, the coproduct over $\mathcal{R}$ is the set of relational terms over $\mathbb{A}$. Hence, a guard $g \in \mathcal{G}$ is a finite conjunction of possibly negated terms of the form $r(a_1, \ldots, a_n)$, $n = \mathsf{ar}(r)$, $a_1, \ldots, a_n \in \mathbb{A}$. Hence, the support of a guard $g \in \mathcal{G}$ is the set of register names $a \in \mathbb{A}$ that appear in it. Then, a register automaton (in the style of [8]) for $\mathcal{R}$ is a tuple (Q, q_0, f, Γ) where

1. Q is a finite supported set of *locations*
2. $q_0\colon 1 \to Q$ is a supported map, the *initial location*,
3. $f\colon Q \to 2$ is a supported map, indicating *finality*,
4. $\Gamma\colon Q \to \mathcal{B}\mathcal{P}_{\mathsf{f}}(\mathcal{G} \times \mathfrak{S}_{\mathsf{f}}(\mathbb{A})\bullet Q)$ is a supported map, the *transitions.*

Here, the properties of supported sets and maps encode precisely the coherence conditions of register automata:

1. Every location has access to finitely many registers.
2. The initial location has all registers uninitialized.
3. There is no side condition on f since 2 is a set.
4. Every transition $q \xrightarrow{g,\pi} q'$ is in such a way that:
   - the guard g may only use registers from $\mathsf{s}(q)$ and the input data value.
   - π tells for each register of q', π its value (from the previous register contents or the input data value).

   If the register contents and the input value satisfy the guard g, then q makes a transition to location q' with the registers rearranged with respect to the injective map

   $$\pi\colon \mathsf{s}_Q(q') \rightarrowtail \{\varrho(0)\} \cup \{\mathsf{old}(a) \mid a \in \mathsf{s}_Q(q)\}$$

   where $\varrho(0)$ is the input value and $\mathsf{old}(\varrho(k)) = \varrho(k+1)$ refers to the old registers contents, i.e. $\mathsf{old}(\varrho(k))$ has the role that $\varrho(k)$ had above the binder $\mathcal{B}$.

In one line, we have an HT-coalgebra on Q for

$$HX = 2 \times \mathcal{B}\mathcal{P}_{\mathsf{f}}(\mathcal{G} \times X) \quad \text{and} \quad TX = \mathfrak{S}_{\mathsf{f}}(\mathbb{A})\bullet X.$$

By Proposition 6.3, H lifts to a functor $G\colon \mathsf{Nom} \to \mathsf{Nom}$, and so Construction 6.1 transforms a register automaton into a nominal set C and an equivariant map

$$c\colon C \longrightarrow 2 \times [\mathbb{A}]\mathcal{P}_{\mathsf{f}}(\mathcal{G} \times C)$$

that is, a nominal automaton.

7 Relation to Presheaves

The category of supported sets $\mathsf{Supp}(\mathbb{A})$ nicely fits into an existing diagram of Kurz, Petrisan, and Velebil [24] that relates nominal sets Nom (for the equality symmetry $\mathfrak{S}_{\mathsf{f}}(\mathbb{A})$) with two presheaf categories:

1. $[\mathbb{I}, \mathsf{Set}]$ is the category of functors $P\colon \mathbb{I} \to \mathsf{Set}$ (*sets in context*), where the objects of $\mathbb{I}$ are finite subsets of $\mathbb{A}$ (i.e. $|\mathbb{I}| = \mathcal{P}_{\mathsf{f}}\mathbb{A}$) and the morphisms are injective maps. This means that for each finite subset $S \subseteq \mathbb{A}$ we have a set PS, and the functoriality of P means that for any injective map $m\colon S \rightarrowtail S'$ ($S' \in \mathcal{P}_{\mathsf{f}}(\mathbb{A})$), we obtain a map $Pm\colon PS \to PS'$. (Note that this a different presheaf category than the one used by Fiore, Plotkin, Turi [12] for variable binding.)
2. $[|\mathbb{I}|, \mathsf{Set}]$ is the category of functors $P\colon |\mathbb{I}| \to \mathsf{Set}$, from the (ordinary) set $\mathcal{P}_{\mathsf{f}}(\mathbb{A})$ to Set, that is, without any functoriality condition. Equivalently, P is a $\mathcal{P}_{\mathsf{f}}(\mathbb{A})$-indexed family of sets.

Figure 2 shows the result when extending the diagram of Kurz et al. [24] by $\mathsf{Supp}(\mathbb{A})$. Let us go through the functors and adjunctions relating the categories (any facts about the adjuctions between Nom, $[|\mathbb{I}|, \mathsf{Set}]$, and $[\mathbb{I}, \mathsf{Set}]$ are recalled from [24]).

1. In the adjunction between $\mathsf{Supp}(\mathbb{A})$ and $[|\mathbb{I}|, \mathsf{Set}]$, the left adjoint R sends a family $X\colon \mathcal{P}_{\mathsf{f}}(\mathbb{A}) \to \mathsf{Set}$ to its coproduct, where the component for $S \in \mathcal{P}_{\mathsf{f}}(\mathbb{A})$ is

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\textbf{Monadic} & \textbf{Monadic} \\
\mathsf{Nom}(\mathfrak{S}_\mathsf{f}(\mathbb{A})) \arrow[r, bend left, "I"] \arrow[r, phantom, "\top"] \arrow[d, bend left, "U"] \arrow[d, phantom, "\dashv"] \arrow[dr, bend left, "D\cdot U"] & {[\mathbb{I}, \mathsf{Set}]} \arrow[l, bend left, "I^*"] \arrow[d, bend left, "U^\mathbb{I}"] \arrow[d, phantom, "\dashv"] \\
\mathsf{Supp}(\mathbb{A}) \arrow[u, bend left, "FX=\mathfrak{S}_\mathsf{f}(\mathbb{A})\bullet X"] \arrow[r, bend left, "D"] \arrow[r, phantom, "\top"] & {[|\mathbb{I}|, \mathsf{Set}]} \arrow[l, bend left, "R"] \arrow[u, bend left, "F^\mathbb{I}"] \arrow[ul, bend left, "F\cdot R"] \arrow[ul, phantom, "\top"]
\end{tikzcd}$$

Figure 2. Adjunctions between supported sets, nominal sets, and presehaf categories.

supported by S:

$$R(X) = \coprod_{S\in\mathcal{P}_\mathsf{f}(\mathbb{A})} X(S) \qquad \mathsf{s}_{RX}(\mathsf{in}_S(x)) = S$$

On a natural transformation $f\colon X \to Y$, the resulting supported map

$$R(f)\colon RX \to RY \quad R(f)(\mathsf{in}_S(x)) = \mathsf{in}_S(f(x))$$

is support-reflecting by definition (hence named R).

2. The right-adjoint $D\colon \mathsf{Supp}(\mathbb{A}) \to [|\mathbb{I}|, \mathsf{Set}]$ forms down-sets: for a supported set X, the family $DX\colon \mathcal{P}_\mathsf{f}\mathbb{A} \to \mathsf{Set}$ is given by

$$DX(S) = \{x \in X \mid \mathsf{s}_X(x) \subseteq S\}$$

For a supported map $f\colon X \to Y$ the natural transformation $Df\colon DX \to DY$ is given by

$$(Df)_S\colon DX(S) \to DY(S) \quad (Df)_S(x) = f(x).$$

3. The composition $D\cdot U\colon \mathsf{Nom} \to [|\mathbb{I}|, \mathsf{Set}]$ sends a nominal set X to a family DUX where for $S \in \mathcal{P}_\mathsf{f}(X)$, $DUX(S) \subseteq X$ contains all elements supported by S. Since adjunctions compose, its left-adjoint is FR. However, this adjunction is not monadic [24], so the monad $DUFR$ does not have Nom as its Eilenberg-Moore category.
4. Instead, $[\mathbb{I}, \mathsf{Set}] = \mathsf{EM}(DUFR)$ [24], and $F^\mathbb{I} \vdash U^\mathbb{I}$ is the corresponding adjunction.
5. The induced comparison functor from Nom is itself adjoint: Nom is (equivalent to) the full reflective subcategory of pullback preserving functors (cf. [15, 19]).

Note that $\mathsf{Supp}(\mathbb{A})$ is the only category in Figure 2 that is not a topos, and therefore not a presheaf.

8 Conclusions and Future Work

We have seen that by going from the base category of sets to supported sets, nominal sets for various symmetries turn out to be monadic. Considering the simple definition of $\mathsf{Supp}(A)$, it is surprising that to the best of our knowledge, it has not been discussed in the literature before.

For name binding, supported sets have a functor inspired by de Bruijn indices, which even lifts to the abstraction functor in nominal sets. It remains for research whether a similar functor for name binding can be found for other data alphabets, most notably for the total order symmetry on $\mathbb{Q}$. Because of the little structure, we conjecture that such a binding functor for $\mathbb{Q}$ can not be found on the level of supported sets.

On the positive side, due to the little structure of supported sets, it provides a common foundation for the nominal sets for different symmetries, in the sense of being described by monads on supported sets. The monadicity can be used to relate nominal automata with register automata, which have a natural definition in supported sets. It remains for future investigation how the *semantics* of register automata can be phrased in supported sets. We are optimistic that it helps to develop a categorical semantics for register automata for data alphabets and signatures beyond symmetries (e.g. those for priority queues [8]). When developing algorithms, in particular learning algorithms, for register or nominal automata [8, 27, 33], supported sets directly yield a finite representation that can help in the implementation.

Acknowledgments

The author thanks Jurriaan Rot, Joshua Moerman, and Frits Vaandrager for inspiring discussions. This work is supported by the NWO TOP project 612.001.852.

References

[1] Jiří Adámek, Stefan Milius, Lurdes Sousa, and Thorsten Wißmann. 2019. Finitely Presentable Algebras for Finitary Monads. *Theory and Applications of Categories* 34, 37 (11 2019), 1179–1195. http://www.tac.mta.ca/tac/volumes/34/37/34-37abs.html

[2] Jiří Adámek, Horst Herrlich, and George E. Strecker. 2004. Abstract and Concrete Categories. The Joy of Cats.

[3] Jiří Adámek and Jiří Rosický. 1994. *Locally Presentable and Accessible Categories.* Cambridge University Press.

[4] Steve Awodey. 2010. *Category Theory.* OUP Oxford. http://books.google.de/books?id=-MCJ6x2lC7oC

[5] Brian E. Aydemir, Aaron Bohannon, and Stephanie Weirich. 2007. Nominal Reasoning Techniques in Coq: (Extended Abstract). *Electron. Notes Theor. Comput. Sci.* 174, 5 (2007), 69–77. https://doi.org/10.1016/j.entcs.2007.01.028

[6] Falk Bartels. 2004. *On generalized coinduction and probabilistic specification formats: Distributive laws in coalgebraic modelling.* Ph.D. Dissertation. Vrije Universiteit Amsterdam.

[7] Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. 2014. Automata theory in nominal sets. *Log. Methods Comput. Sci.* 10 (2014). https://doi.org/10.2168/LMCS-10(3:4)2014

[8] Sofia Cassel, Falk Howar, Bengt Jonsson, and Bernhard Steffen. 2018. Extending Automata Learning to Extended Finite State Machines. In *Machine Learning for Dynamic Software Analysis: Potentials and Limits - International Dagstuhl Seminar 16172, Dagstuhl Castle, Germany, April 24-27, 2016, Revised Papers (Lecture Notes in Computer Science, Vol. 11026)*, Amel Bennaceur, Reiner Hähnle, and Karl Meinke (Eds.). Springer, 149–177. https://doi.org/10.1007/978-3-319-96562-8_6

[9] Vincenzo Ciancia and Ugo Montanari. 2008. A Name Abstraction Functor for Named Sets. In *Proceedings of the Ninth Workshop on*

Coalgebraic Methods in Computer Science, CMCS 2008, Budapest, Hungary, April 4-6, 2008 (Electronic Notes in Theoretical Computer Science, Vol. 203), Jirí Adámek and Clemens Kupke (Eds.). Elsevier, 49–70. https://doi.org/10.1016/j.entcs.2008.05.019

[10] N.G de Bruijn. 1972. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)* 75, 5 (1972), 381–392. https://doi.org/10.1016/1385-7258(72)90034-0

[11] Stéphane Demri and Ranko Lazic. 2009. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.* 10, 3 (2009), 16:1–16:30. https://doi.org/10.1145/1507244.1507246

[12] Marcelo Fiore, Gordon Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding (Extended Abstract). In *Proc. 14$^{\text{th}}$ LICS Conf.* IEEE, Computer Society Press, 193–202.

[13] Marcelo P. Fiore and Sam Staton. 2006. Comparing operational models of name-passing process calculi. *Inf. Comput.* 204, 4 (2006), 524–560. https://doi.org/10.1016/j.ic.2005.08.004

[14] Murdoch Gabbay and James Cheney. 2004. A Sequent Calculus for Nominal Logic. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings.* IEEE Computer Society, 139–148. https://doi.org/10.1109/LICS.2004.1319608

[15] Murdoch Gabbay and Andrew M. Pitts. 1999. A New Approach to Abstract Syntax Involving Binders. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999.* IEEE Computer Society, 214–224. https://doi.org/10.1109/LICS.1999.782617

[16] Murdoch Gabbay and Andrew M. Pitts. 2002. A New Approach to Abstract Syntax with Variable Binding. *Formal Aspects Comput.* 13, 3-5 (2002), 341–363. https://doi.org/10.1007/s001650200016

[17] Murdoch James Gabbay and Martin Hofmann. 2008. Nominal Renaming Sets. In *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008, Doha, Qatar, November 22-27, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 5330)*, Iliano Cervesato, Helmut Veith, and Andrei Voronkov (Eds.). Springer, 158–173. https://doi.org/10.1007/978-3-540-89439-1_11

[18] Peter Gabriel and Friedrich Ulmer. 1971. *Lokal präsentierbare Kategorien.* Lecture Notes Math., Vol. 221. Springer-Verlag.

[19] Fabio Gadducci, Marino Miculan, and Ugo Montanari. 2006. About Permutation Algebras, (Pre)Sheaves and Named Sets. *Higher Order Symbol. Comput.* 19, 2-3 (September 2006), 283–304. https://doi.org/10.1007/s10990-006-8749-3

[20] Peter T. Johnstone. 1975. Adjoint Lifting Theorems for Categories of Algebras. *Bull. London Math. Soc.* 7, 3 (1 nov 1975), 294–297.

[21] Michael Kaminski and Nissim Francez. 1994. Finite-Memory Automata. *Theor. Comput. Sci.* 134, 2 (1994), 329–363. https://doi.org/10.1016/0304-3975(94)90242-9

[22] Dexter Kozen, Konstantinos Mamouras, Daniela Petrisan, and Alexandra Silva. 2015. Nominal Kleene Coalgebra. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9135)*, Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (Eds.). Springer, 286–298. https://doi.org/10.1007/978-3-662-47666-6_23

[23] Alexander Kurz, Daniela Petrisan, Paula Severi, and Fer-Jan de Vries. 2013. Nominal Coalgebraic Data Types with Applications to Lambda Calculus. *Logical Methods in Computer Science* 9, 4 (2013).

[24] Alexander Kurz, Daniela Petrisan, and Jiri Velebil. 2010. Algebraic Theories over Nominal Sets. *CoRR* abs/1006.3027 (2010). http://arxiv.org/abs/1006.3027

[25] Saunders Mac Lane. 1998. *Categories for the Working Mathematician.* Springer New York. http://books.google.de/books?id=eBvhyc4z8HQC

[26] Joshua Moerman and Jurriaan Rot. 2020. Separation and Renaming in Nominal Sets. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 152)*, Maribel Fernández and Anca Muscholl (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 31:1–31:17. https://doi.org/10.4230/LIPIcs.CSL.2020.31

[27] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michal Szynwelski. 2017. Learning nominal automata. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, Giuseppe Castagna and Andrew D. Gordon (Eds.). ACM, 613–625. https://doi.org/10.1145/3009837.3009879

[28] Andrew M. Pitts. 2013. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge Tracts in Theoretical Computer Science, Vol. 57. Cambridge University Press.

[29] Lutz Schröder, Dexter Kozen, Stefan Milius, and Thorsten Wißmann. 2017. Nominal Automata with Name Binding. In *Proc. 20th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2017) (LNCS, Vol. 10203)*, Javier Esparza and Andrzej Murawski (Eds.). Springer, 124–142. https://doi.org/10.1007/978-3-662-54458-7_8

[30] Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. 2013. Generalizing determinization from automata to coalgebras. *Log. Methods Comput. Sci.* 9, 1 (2013). https://doi.org/10.2168/LMCS-9(1:9)2013

[31] Sam Staton. 2007. *Name-passing process calculi: operational models and structural operational semantics.* Technical Report UCAM-CL-TR-688. University of Cambridge, Computer Laboratory. https://doi.org/10.48456/tr-688

[32] Christian Urban and Christine Tasson. 2005. Nominal Techniques in Isabelle/HOL. In *Automated Deduction - CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3632)*, Robert Nieuwenhuis (Ed.). Springer, 38–53. https://doi.org/10.1007/11532231_4

[33] Henning Urbat and Lutz Schröder. 2020. Automata Learning: An Algebraic Approach. In *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 900–914. https://doi.org/10.1145/3373718.3394775

[34] David Venhoek, Joshua Moerman, and Jurriaan Rot. 2018. Fast Computations on Ordered Nominal Sets. In *Theoretical Aspects of Computing - ICTAC 2018 - 15th International Colloquium, Stellenbosch, South Africa, October 16-19, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11187)*, Bernd Fischer and Tarmo Uustalu (Eds.). Springer, 493–512. https://doi.org/10.1007/978-3-030-02508-3_26

[35] David Venhoek, Joshua Moerman, and Jurriaan Rot. 2019. Fast Computations on Ordered Nominal Sets. *CoRR* abs/1902.08414 (2019). arXiv:1902.08414 http://arxiv.org/abs/1902.08414

A Omitted Proofs

Details for Example 2.4

All examples listed here satisfy the M-set axioms:

1. A is an M-set because indeed $\mathsf{id}_A \cdot a = \mathsf{id}_A(a) = a$ and $(\ell \cdot m) \cdot a = \ell(m(a)) = \ell \cdot (m \cdot a)$.

2./3. The verification of the M-set axioms for $\mathcal{P}_\mathsf{f}(X)$ and discrete M-sets are straightforward.

4. The map $h\colon M \to M^M$ is a monoid-homomorphism because:

$$\begin{aligned} h(\mathsf{id}_A) &= (\ell \mapsto \mathsf{id}_A \cdot \ell) = (\ell \mapsto \ell) = \mathsf{id}_M \\ h(m \cdot n) &= (\ell \mapsto (m \cdot n) \cdot \ell) = (\ell \mapsto m \cdot (n \cdot \ell)) \\ &= (\ell' \mapsto m \cdot \ell') \cdot (\ell \mapsto n \cdot \ell) = h(m) \cdot h(n) \qquad \square \end{aligned}$$

Proof of Lemma 2.7

By definition, $\approx_S$ is reflexive, symmetric, and transitive.

We now verify that $m \approx_S m'$ implies $\ell \cdot m \approx_S \ell \cdot m'$ for all $\ell, m, m' \in M$. If $m \approx_S m'$, then $m(a) = m'(a)$ for all $a \in S$. Hence, $\ell(m(a)) = \ell(m'(a))$ for all $a \in S$, i.e. $\ell \cdot m \approx_S \ell \cdot m'$. □

Proof of Lemma 2.9

For 'only if', put $m' := \mathsf{id}_A$. For 'if', take $m, m' \in M$ with $m(a) = m'(a)$ for all $a \in S$. Hence, $(m'^{-1} \cdot m)(a) = a$ for all $a \in S$, hence $m \cdot x = m' \cdot m'^{-1} \cdot m \cdot x = m' \cdot x$. □

Details for Example 2.10

The examples of nominal M-sets Example 2.10 are all standard. It only remains to verify the non-example in item 4, i.e. let us verify that whenever M is a nominal M-set, then all M-sets are nominal.

If M is a nominal M-set, let $S \subseteq A$ be a finite set that supports $\mathsf{id}_A \in M$. In fact, S is the support of all elements $x \in X$ in all M-sets X: consider $m, m' \in M$ with $m \approx_S m'$. Then $m = m \cdot \mathsf{id}_A = m' \cdot \mathsf{id}_A = m'$ because S supports id_A. Hence, $m \cdot x = m' \cdot x$, showing that S also supports $x \in X$. □

Proof of Lemma 2.14

If S supports x, consider $m, m' \in M$ with $m \approx_S m'$. Then, $m \cdot f(x) = f(m \cdot x) = f(m' \cdot x) = m' \cdot f(x)$.

For least finite supports, we have that $\mathsf{supp}_X(x)$ supports $f(x)$ by the previous statement. Hence, the least finite support $\mathsf{supp}_Y(f(x))$ is smaller or equal to $\mathsf{supp}_X(x)$ (w.r.t. subset inclusion). □

Proof of Lemma 2.15

The proof for $M := \mathsf{Fin}(A)$ can be found in [17, Lem. 11] and straightforwardly adapts to arbitrary monoids M:

1. We show that $m[\mathsf{supp}_X(x)]$ is a support of $m \cdot x$. Hence, consider $k, k' \in M$ with $k \approx_{m[\mathsf{supp}_X(x)]} k'$. Thus, $k \cdot m \approx_{\mathsf{supp}_X(x)} k' \cdot m$ and so $(k \cdot m) \cdot x = (k' \cdot m) \cdot x$ (since $\mathsf{supp}_X(x)$ supports x). So $k \cdot (m \cdot x) = k' \cdot (m \cdot x)$, showing that $m[\mathsf{supp}_X(x)]$ supports $m \cdot x$.

2. Let $m^{-1} \in M$ be the (left and right) inverse of m. Then we verify the other inclusion direction:

$$\begin{aligned} m \cdot \mathsf{supp}_X(x) &= m \cdot \mathsf{supp}_X(m^{-1} \cdot m \cdot x) \\ &\subseteq m \cdot m^{-1} \cdot \mathsf{supp}_X(m \cdot x) \quad \text{(by the item 1)} \\ &= \mathsf{supp}_X(m \cdot x) \qquad \square \end{aligned}$$

Proof of Lemma 3.3

For arbitrary $M \le A^A$, define the functor $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ sending $(X, \cdot)$ to the set X with the support

$$\mathsf{s}_X(x) = \bigcap\{S \subseteq A \mid S \text{ finite and supports } x\}$$

Since every element $x \in X$ has some finite support (Definition 2.8), the intersection in the definition of s_X is not empty, and hence $\mathsf{s}_X(x)$ is a finite set and so U sends nominal M-sets to supported sets.

For an equivariant map $f\colon X \to Y$ and $x \in X$, we have by Lemma 2.14:

$$\begin{aligned} &\{S \subseteq A \mid S \text{ finite and supports } f(x)\} \\ &\quad \supseteq \{S \subseteq A \mid S \text{ finite and supports } x\}. \end{aligned}$$

When taking the intersection of these families of subsets of A, the inclusion reverses and we obtain

$$\begin{aligned} \mathsf{s}_Y(f(x)) &= \bigcap\{S \subseteq A \mid S \text{ finite and supports } f(x)\} \\ &\subseteq \bigcap\{S \subseteq A \mid S \text{ finite and supports } x\} = \mathsf{s}_X(x) \end{aligned}$$

Thus, Uf is a supported map. Clearly, U preserves identities and composition. Note that we do did not require here that *least* finite supports exist in nominal M-sets.

If they exist however, we have that the least finite support $\mathsf{supp}_X(x)$ of $x \in X$ is the intersection of all finite supports of x, i.e. $\mathsf{s}_X(x) = \mathsf{supp}_X(x)$, as claimed. □

Proof of Lemma 3.4

Define $J\colon \mathsf{Set} \hookrightarrow \mathsf{Supp}(A)$ by $JX = (X, \mathsf{s}_X)$, $\mathsf{s}_X(x) = \emptyset$. Every map $f\colon X \to Y$ then is a supported map $f\colon JX \to JY$. Clearly, $UJX = X$ and $UJ(f\colon X \to Y) = f$, hence $UJ = \mathsf{Id}_{\mathsf{Set}}$.

For the adjunction, we verify that there are supported maps $\eta_X\colon X \to JUX$ such that for every supported map $h\colon X \to JY$ ($Y \in \mathsf{Set}$), there is a unique $h'\colon UX \to Y$ with $h = Jh' \cdot \eta_X$:

$$\begin{tikzcd} JUX \arrow[r, "Jh'"] & JY \\ X \arrow[u, "\eta_X"] \arrow[ur, "h"'] & \end{tikzcd}$$

Note that the identity $\mathsf{id}_X\colon X \to X$ gives rise to the supported map $\eta_X\colon X \to JUX$. Hence, $Jh' \cdot \eta_X = h$ and that $h' := Uh$. Since η_X is surjective and J is faithful, h' is the unique map fulfilling this property. □

Proof of Proposition 3.5

Consider a diagram $D\colon \mathcal{D} \to \mathrm{Supp}(A)$ and its colimiting cocone in Set:

$$\mathsf{in}_Y\colon DY \to C \qquad \text{in Set.}$$

In order to make C a supported set, define $\mathsf{s}\colon C \to \mathcal{P}_\mathrm{f}(A)$ by

$$\mathsf{s}(c) = \bigcap \{\mathsf{s}_{DY}(y) \mid Y \in \mathcal{D}, y \in DY, \mathsf{in}_Y(y) = c\}$$

Since colimit injections in Set are jointly surjective, the above intersection is non-empty, and thus yields a finite subset of A.

This lets the cocone lift to $\mathrm{Supp}(A)$: every $\mathsf{in}_Z\colon DZ \to C$ is a supported map, because for every $z \in DZ$ we have

$$\begin{aligned}\mathsf{s}_C(\mathsf{in}_Z(z)) &= \bigcap \{\mathsf{s}_{DY}(y) \mid Y \in \mathcal{D}, y \in DY, \mathsf{in}_Y(y) = \mathsf{in}_Z\} \\ &\subseteq \mathsf{s}_{DZ}(z).\end{aligned}$$

In order to see that $(\mathsf{in}_Y)_{Y\in\mathcal{D}}$ is a colimiting cocone, consider a cocone $(e_Y\colon DY \to E)_{Y\in\mathcal{D}}$ in $\mathrm{Supp}(A)$. Since this is also a competing cocone in Set, we obtain a cocone morphism $u\colon C \to E$. For the verification that u is a supported map, note that for every $c \in C$ and every $Y \in \mathcal{D}$, $y \in DY$ with $\mathsf{in}_Y(y) = c$, we have $\mathsf{s}_E(u(c)) \subseteq \mathsf{s}_Y(y)$. By the universal property of intersection, this implies that

$$\mathsf{s}_E(u(c)) \subseteq \bigcap \{\mathsf{s}_Y(y) \mid Y \in \mathcal{D}, y \in DY, \mathsf{in}_Y(y) = c\} = \mathsf{s}_C(c),$$

so $u\colon C \to E$ is a supported map. Since $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$ is faithful, u is a cocone morphism in $\mathrm{Supp}(A)$ and moreover unique. □

Verification of Example 3.6

We show the statement about coproducts directly for I-indexed families $(X_i)_{i\in I}$: by Proposition 3.5, the coproduct $\coprod_{i\in I} X_i$ in $\mathrm{Supp}(A)$ is given by the coproduct in Set. For every $c \in \coprod_{i\in I} X_i$ there is precisely one $i \in I$, $x \in X_i$ with $\mathsf{in}_i(x) = c$. Hence, $\mathsf{s}_{\coprod_{i\in I} X_i}(c) = \mathsf{s}_{X_i}(x)$.

With this characterization of coproducts, every supported set Y, is the coproduct of singleton sets

$$Y = \coprod_{y\in Y} \{y\} \qquad \text{where } \mathsf{s}_{\{y\}}(y) = \mathsf{s}_Y(y) \qquad \square$$

Proof of Proposition 3.7

Consider a diagram $D\colon \mathcal{D} \to \mathrm{Supp}(A)$ and its limiting cone in Set:

$$\mathsf{pr}_Y\colon P \to DY \qquad \text{in Set.}$$

Define the map to all – i.e. possibly infinite – subsets of A:

$$p\colon P \to \mathcal{P}(A) \qquad p(x) = \bigcup \{\mathsf{s}_{DY}(\mathsf{pr}_Y(x)) \mid Y \in \mathcal{D}\}$$

The limit in $\mathrm{Supp}(A)$ is the restriction of P to finitely supported elements:

$$L := \{x \in P \mid p(x) \text{ is finite}\} \qquad \mathsf{s}_L(x) := p(x).$$

By definition, L is a finitely supported map. The projections are the restrictions of pr_Y to $L \subseteq P$: $\ell_Y\colon L \to DY$ and $\ell(x) = \mathsf{pr}_Y(x)$ for all $Y \in \mathcal{D}$. Hence,

$$\mathsf{s}_L(x) = \bigcup \{\mathsf{s}_{DY}(\ell_Y(x)) \mid Y \in \mathcal{D}\}.$$

It is immediate that every ℓ_Y is a supported map:

$$\mathsf{s}_{DY}(\ell_Y(x)) \subseteq \bigcup \{\mathsf{s}_Z(\ell_Z(x)) \mid Z \in \mathcal{D}\} = \mathsf{s}_L(x)$$

The family $(\ell_Y\colon L \to DY)_{Y\in\mathcal{D}}$ is a cone because it was defined as a restriction of the cone P in Set. For the verification of the universal property, consider another cone

$$(e_Y\colon E \to DY)_{Y\in\mathcal{D}} \qquad \text{in } \mathrm{Supp}(A).$$

This is also a cone for the diagram $UD\colon \mathcal{D} \to \mathrm{Set}$, so we obtain a unique cone morphism $u\colon E \to P$ (i.e. $\mathsf{pr}_Y \cdot u = e_Y$ for all $Y \in \mathcal{D}$). For every $x \in E$ and $Y \in \mathcal{D}$, we have

$$\mathsf{s}_{DY}(\mathsf{pr}_Y(u(x))) = \mathsf{s}_{DY}(e_Y(x)) \subseteq \mathsf{s}_E(x)$$

since e_Y is a supported map. So by the universal property of union, we obtain

$$p(u(x)) = \bigcup \{\mathsf{s}_{DY}(\mathsf{pr}_Y(u(x))) \mid Y \in \mathcal{D}\} \subseteq \mathsf{s}_E(x),$$

which proves the finiteness of $p(u(x)) \subseteq A$. Hence, $u\colon E \to P$ restricts to $u\colon E \to L$ which is thus also a supported map ($\mathsf{s}_L(u(x)) = p(u(x)) \subseteq s_E(x)$). Uniqueness of $u\colon U \to L$ follows directly from the faithfulness of $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$. Hence, $(\ell_Y\colon L \to DY)_{Y\in\mathcal{D}}$ is a limiting cone. □

Proof of Corollary 3.8

Consider a diagram $D\colon \mathcal{D} \to \mathrm{Supp}(A)$ and its limit

$$(\mathsf{pr}_Y\colon \lim UD \to UDY)_{Y\in\mathcal{D}} \qquad \text{in Set.}$$

If $\mathcal{D}$ has only finitely many objects, then

$$\bigcup \{\mathsf{s}_{DY}(\mathsf{pr}_Y(x)) \mid Y \in \mathcal{D}\}$$

is finite for all $x \in \lim UD$. Hence, the limit in $\mathrm{Supp}(A)$ given by Proposition 3.7 is identical to the limit $\lim UD$ in Set. □

Proof of Proposition 3.11

Define $\eta_X\colon X \to (X \times E)^E$ by

$$\eta_X(x)(e) = (x, e).$$

For every $x \in X$, clearly $\eta_X(x)$ is in $(X \times E)^E$ since

$$\mathsf{s}_{(X\times E)^E}(\eta_X(x)) = \bigcup_{e\in E} \mathsf{s}_{X\times E}(x, e) \setminus \mathsf{s}_E(e) = \bigcup_{e\in E} \mathsf{s}_X(x) = \mathsf{s}_X(x)$$

is finite; this also shows that η_X is a supported map. Moreover, η_X is natural in X since

$$\begin{aligned}(g \times E)^E(\eta_X(x) &= (g \times E) \cdot (e \mapsto (x, e)) \\ &= (e \mapsto (g(x), e) = \eta_Y(g(x)).\end{aligned}$$

For the verification of the universal mapping property, consider $g\colon X \to Y^E$ and define

$$h\colon X \times E \to Y \quad \text{by} \quad h(x, e) = g(x)(e).$$

By the definition of s_{Y^E}, we have

$$\mathsf{s}_Y(g(x)(e)) \setminus \mathsf{s}_E(e) \quad\subseteq\quad \mathsf{s}_{Y^E}(g(x))$$

and so

$$\mathsf{s}_Y(g(x)(e)) \subseteq \mathsf{s}_{Y^E}(g(x)) \cup \mathsf{s}_E(e) \subseteq \mathsf{s}_X(x) \cup \mathsf{s}_E(e) = \mathsf{s}_{X\times E}(x,e),$$

hence, h is a supported map. For all $x \in X$, we have

$$\begin{aligned} h^E(\eta_X(x)) &= h\cdot(\eta(x)) = h\cdot(e\mapsto(x,e)) \\ &= (e\mapsto h(x,e)) = (e\mapsto g(x)(e)) = g(x), \end{aligned}$$

so $h^E\cdot\eta_X = g$. For uniqueness, consider a supported map $u\colon X\times E\to Y$ with $u^E\cdot\eta_X = g$. Necessarily,

$$g(x) = u^E(\eta_X(x)) = u\cdot(e\mapsto(x,e)) = (e\mapsto u(x,e))$$

i.e. $u(x,e) = g(x)(e) = h(x,e)$ and $u = h$. □

Proof of Proposition 3.12

Let us first recall the definition of fp objects and lfp categories:

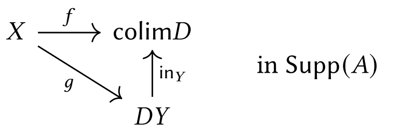
Definition A.1 [3, 18].
1. A *directed colimit* is the colimit for a diagram $D\colon \mathcal{D}\to \mathcal{C}$ where $\mathcal{D}$ is a poset in which any two objects have an upper bound.
2. An object X in a category $\mathcal{C}$ is called *finitely presented (fp)* if the hom-functor $\mathcal{C}(X,-)\colon \mathcal{C}\to \mathsf{Set}$ preserves colimits of directed diagrams.
3. A category $\mathcal{C}$ is called *locally finitely presentable (lfp)* if it is cocomplete and every object is the directed colimit of finitely presentable objects.

Given $X\in\mathsf{Supp}(A)$ and a directed diagram $D\colon\mathcal{D}\to\mathsf{Supp}(A)$, note that the colimit preservation

$$\mathsf{Supp}(A)(X,\mathsf{colim}D) = \mathsf{colim}\mathsf{Supp}(A)(X,D(-))$$

boils down to the following condition [3, Def. 1.1]: for every supported map $f\colon X\to\mathsf{colim}D$, there exists $Y\in\mathcal{D}$ such that

1. there are $Y\in\mathcal{D}$ and $g\colon X\to DY$ with $\mathsf{in}_Y\cdot f' = f$:

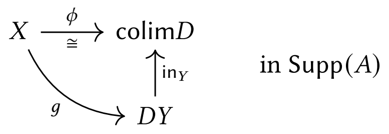


$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "g"'] & \mathsf{colim}D \\ & DY \arrow[u, "\mathsf{in}_Y"'] \end{tikzcd} \quad \text{in } \mathsf{Supp}(A)$$

2. Y and g are essentially unique in the sense that if $f = \mathsf{in}_{Y'}\cdot g'$ for $g'\colon X\to DY'$, then $D(Y\to Z)\cdot g = D(Y'\to Z)\cdot g'$ for some $Z\geq Y$ in $\mathcal{D}$ (recall that $\mathcal{D}$ is a poset, so we can denote connecting morphisms simply by $Y\to Z$ and $Y'\to Z$)

We now prove the following statements:

1. Every supported set X is the directed colimit of its finite subsets:
Let $\mathcal{D} = \{Y \mid Y\subseteq X\}$, $D\colon\mathcal{D}\to\mathsf{Supp}(A)$, $DY = (Y,\mathsf{s}_{DY})$ where s_{DY} is the restriction of $\mathsf{s}_X\colon X\to\mathcal{P}_\mathsf{f}(A)$ to $Y\subseteq X$. In $\mathcal{D}$, any two elements $Y,Y'\in\mathcal{D}$ have an upper bound: $Y\cup Y'\in\mathcal{D}$. By Proposition 3.5, we have that $\mathsf{colim}D$ is carried by X, and
$$\begin{aligned}\mathsf{s}_{\mathsf{colim}D}(x) &= \bigcap\{\mathsf{s}_{DY}(x) \mid Y\in\mathcal{D}, x\in Y\} \\ &= \bigcap\{\mathsf{s}_X(x)\mid Y\in\mathcal{D}, x\in Y\} = \mathsf{s}_X(x).\end{aligned}$$
So $\mathsf{colim}D\cong X$, as desired.
2. Every finitely presentable supported set X is finite:
Consider the directed colimit of subsets of X. By item 1, this directed colimit yields X, so we have an isomorphism $\phi\colon X\to\mathsf{colim}D$. Since X is finitely presentable, this factors through some finite subset $Y\subseteq X$:



$$\begin{tikzcd} X \arrow[r, "\phi", "\cong"'] \arrow[dr, bend right, "g"'] & \mathsf{colim}D \\ & DY \arrow[u, "\mathsf{in}_Y"'] \end{tikzcd} \quad \text{in } \mathsf{Supp}(A)$$

Note that in_Y is support-preserving by the definition of $\mathcal{D}$ and injective since it's the same colimit injection as in Set. Also, in_Y is surjective, since $\mathsf{in}_Y\cdot g = \phi$. Hence, it is a regular monomorphism, an epimorphism and thus an isomorphism, showing finiteness of X.
3. Every finite supported set X is finitely presentable:
Since finitely presentable objects are closed under finite coproducts [3, Prop. 1.3] and every finite set is the finite coproduct of singleton sets (Example 3.6), we can assume wlog that X is singleton $X=\{x\}$. For the verification that X is fp, consider a supported map $f\colon X\to\mathsf{colim}D$ for a directed diagram $D\colon\mathcal{D}\to\mathsf{Supp}(A)$. Since colimit injections are jointly surjective in Set and thus also in $\mathsf{Supp}(A)$, there are $Z\in\mathcal{D}$, $z\in DZ$ with $\mathsf{in}_Z(z) = f(x)$. However, the support of z could possibly contain more than $\mathsf{s}(f(x))$. Denote this difference by the finite set
$$R := \mathsf{s}_{DZ}(z)\setminus\mathsf{s}_{\mathsf{colim}D}(f(x)) \qquad \subseteq A.$$
   - If $R := \emptyset$, then $\mathsf{s}_{DZ}(z) = \mathsf{s}_{\mathsf{colim}D}(f(x))\subseteq\mathsf{s}_X(x)$ and we have the desired supported map $g\colon X\to DZ$, $g(x) = z$.
   - By the characterization of colimits in $\mathsf{Supp}(A)$ (Proposition 3.5), we know that
   $$\mathsf{s}_{\mathsf{colim}D}(f(x)) = \bigcap\{\mathsf{s}_Y(y)\mid Y\in\mathcal{D}, \mathsf{in}_Y(y) = f(x)\}$$
   So for every $a\in R$, there must be some $Y_a\in\mathcal{D}$ and $y_a\in DY_a$ with
   $$\mathsf{in}_{Y_a}(y_a) = f(x) \text{ and } a\notin\mathsf{s}_{DY_a}(y_a).$$
   Let $U\in\mathcal{D}$ be the upper bound of $\{Y_a\mid a\in R\}$ in $\mathcal{D}$, which exists by assumption. Then $u := D(Y_a\to U)(y_a)$ fulfils
   $$\mathsf{s}_U(u) = \mathsf{s}_{\mathsf{colim}D}(f(x))\subseteq\mathsf{s}_X(x) \text{ and } \mathsf{in}_U(u) = f(x)$$
   so we have the desired supported map $g\colon X\to U$, $g(x) = u$.

The essential uniqueness of the factorization follows on the level of Set: consider two factorizations:

$$g\colon X \to DY \quad g'\colon X \to DY' \quad \mathsf{in}_Y \cdot g = f = \mathsf{in}_{Y'} \cdot g'$$

Since $g(x)$ and $g'(x)$ are identified in the colimit ($\mathsf{in}_Y(g(x)) = \mathsf{in}_{Y'}(g'(x))$) and since $\mathcal{D}$ is directed, by [3, Ex. 1a.2.ii] there is some $Z \in \mathcal{D}$ with $Z \geq Y, Z \geq Y'$ and

$$D(Y \to Z)(g(x)) = D(Y' \to Z)(g'(x)).$$

4. Since $\mathrm{Supp}(A)$ is cocomplete, and since we have characterized the finitely presentable objects as the finite supported sets, item 1 shows that $\mathrm{Supp}(A)$ is lfp. □

Proof of Lemma 3.13

- Epi = surjective: Every surjective supported map is an epimorphism because the forgetful $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$ is faithful. Conversely, if $e\colon X \to Y$ is an epimorphism, assume that there was some $y \in Y$ not in the image of e. Consider the maps $f, g\colon Y \to Y + 2$ where we extend Y by two elements $2 = \{0, 1\}$ with empty support and put $f(y) = 0$ and $g(y) = 1$ and $f(z) = g(z) = z$ for all $z \in Y \setminus \{y\}$. By the assumption of $y \notin \mathsf{Im}(e)$, $f \cdot e = g \cdot e$, and so $f = g$, a contradiction.
- Mono = injective: Every injective supported map is a monomorphism because the forgetful $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$ is faithful. Conversely, for a monomorphism $m\colon X \to Y$ in $\mathrm{Supp}(A)$, consider $a, b \in X$ with $m(x) = m(y)$. Define

$$Z = 1 = \{0\} \text{ with } \mathsf{s}_Z(0) = \mathsf{s}_X(a) \cup \mathsf{s}_X(b),$$

hence we have supported maps $f_a, f_b\colon Z \to X$ with $f_a(0) = a$, $f_b(0) = b$. Thus, $m \cdot f_a = m \cdot f_b$. Since m is monic, we obtain $f_a = f_b$ and so $a = b$ as desired. □

Proof of Proposition 3.16

If a bijective map $f\colon X \to Y$ is support reflecting, then $f^{-1}\colon Y \to X$ fulfils $\mathsf{s}_Y = \mathsf{s}_X \cdot f^{-1}$ and hence f^{-1} is a supported map; thus f is an isomorphism.

Conversely, if $f\colon X \to Y$ is an isomorphism, then f is bijective and we have a supported map $f^{-1}\colon Y \to X$ with

$$\mathsf{s}_Y(f(x)) \supseteq \mathsf{s}_X(f^{-1}(f(x))) = \mathsf{s}_X(x).$$

Thus, $\mathsf{s}_Y \cdot f = \mathsf{s}_X$ in total, i.e. f is support-reflecting. □

Proof of Proposition 3.17

Recall that an epimorphism $e\colon X \to Y$ is called regular if it is a coequalizer.

Assume that $e\colon X \to Y$ is a coequalizer of f and g, in particular $e \cdot f = e \cdot g$:

$$\begin{tikzcd} R \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & X \arrow[r, "e"] & Y \end{tikzcd}$$

By the characterization of colimits (Proposition 3.5), we directly obtain

$$\begin{aligned} \mathsf{s}_Y(y) = {} & \bigcap\{\mathsf{s}_R(r) \mid r \in R, e(f(r)) = y\} \\ & \cap \bigcap\{\mathsf{s}_X(x) \mid x \in X, e(x) = y\} \end{aligned}$$

For every $r \in R$ with $e(f(r)) = y$, we have $\mathsf{s}_X(f(r)) \subseteq \mathsf{s}_R(r)$ and so the first conjunct disappears:

$$\mathsf{s}_Y(y) = \bigcap\{\mathsf{s}_X(x) \mid x \in X, e(x) = y\}$$ □

Proof of Proposition 3.18

We prove both statements together via the following steps:

1. There is a subobject classifier for support-preserving monomorphisms.
2. Every regular monomorphism is support-preserving.
3. Every support-preserving monomorphism is regular.

For the verification:

1. For a support-preserving monomorphism $m\colon S \to X$, define $\chi_S\colon X \to 2$ to be the usual characteristic function, i.e. the subobject classifier in Set:

$$\chi_S(x) = \begin{cases} 1 & \text{if there is some } s \in S \text{ with } m(s) = x \\ 0 & \text{otherwise.} \end{cases}$$

Hence, $\chi_S \cdot m = t \cdot !$ since the faithful $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$ sends this to the commutative square of the subobject classifier in Set.
For the verification of the universal property of the pullback, consider a cone (C, c, d), i.e. supported maps $!_C\colon C \to 1$, $d\colon C \to X$ with $\chi_S \cdot d = t \cdot !_C$.

$$\begin{tikzcd} C \arrow[dr, dashed, "u"] \arrow[drr, bend left, "d"] \arrow[ddr, bend right, "!_C"'] & & \\ & S \arrow[r, "m"] \arrow[d, "!"'] & X \arrow[d, "\chi_S"] \\ & 1 \arrow[r, "t"] & 2 \end{tikzcd}$$

The universal property of the subobject classifier in Set induces a map $u\colon C \to S$ with $m \cdot u = d$ and $! \cdot u = !_C$. All we need to verify is that u is a supported map. Since m is support-preserving, we have $\mathsf{s}_X \cdot m = \mathsf{s}_S$ and thus for all $y \in C$

$$\mathsf{s}_S(u(y)) = \mathsf{s}_X(m(u(y))) = \mathsf{s}_X(d(y)) \subseteq \mathsf{s}_C(y)$$

as required.
The uniqueness of u is clear because $U\colon \mathrm{Supp}(A) \to \mathrm{Set}$ is faithful.

2. Consider a regular monomorphism $m\colon S \to X$, that is, the equalizer of a parallel pair of morphisms $f, g\colon X \to Y$:

$$\begin{tikzcd} S \arrow[r, "m"] & X \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & Y \end{tikzcd}$$

By Proposition 3.7, the support on the limit S is

$$\mathsf{s}_S(z) \overset{(1)}{=} \mathsf{s}_X(m(z)) \cup \mathsf{s}_Y(f(m(z))) = \mathsf{s}_X(m(z))$$

where we use that f is a supported map in the second equation.

3. Given a support-preserving monomorphism $m\colon S \to X$, we have a pullback square

$$\begin{tikzcd} S \arrow[r, "m"] \arrow[d, "!"'] & X \arrow[d, "\chi_S"] \arrow[dl, "!"'] \\ 1 \arrow[r, "t"] & 2 \end{tikzcd}$$

Since 1 is the terminal object, we have that m is the equalizer of χ_S and $t\cdot!$:

$$\begin{tikzcd} S \arrow[r, "m"] & X \arrow[r, shift left, "\chi_S"] \arrow[r, shift right, "t\cdot !"'] & 2 \end{tikzcd}$$

(Note that this is just the argument that if a monomorphism m has a subobject classifier, then it is a regular monomorphism). □

Proof of Lemma 4.4

Since $\approx_S$ is a congruence for left-multiplication (Lemma 2.7), the M-set structure on M/S by $\ell \cdot [m]_S := [\ell \cdot m]_S$ is well-defined. For the verification that $m[S]$ supports $[m]_S$, take $\ell, \ell' \in M$ with $\ell \approx_{m[S]} \ell'$. For all $a \in S$ we have $m(a) \in m[S]$ and so $\ell(m(a)) = \ell'(m(a))$. Thus, $[\ell \cdot m]_S = [\ell' \cdot m]_S$ and

$$\ell \cdot [m]_S = [\ell \cdot m]_S = [\ell' \cdot m]_S = \ell \cdot [m]_S. \qquad □$$

Proof of Example 4.6

For plain Nom and renaming nominal sets, one simply picks some $b \in A$ fresh for R and $\{a\}$ and puts $\ell = (a\,b)$, i.e. $\ell(a) = b$, $\ell(b) = a$, and $\ell(x) = x$ for $x \notin \{a, b\}$.

For the order symmetry, a more involved argument is necessary. Venhoek et al. [35, Lemma 5.2] show a property called *homogeneity*: for any two finite $C \subseteq \mathbb{Q}$, $C \subseteq \mathbb{Q}'$, if $|C| = |C'|$ then there is a $\pi \in \mathrm{Aut}(\mathbb{Q}, <)$ with $\pi[C] = C'$.

Apply this for

$$b := a + \frac{1}{2} \cdot \min_{x \in R} |a - x| \qquad C := R \cup \{a\} \qquad C' := R \cup \{b\}$$

Since $a \notin R$, we have $b \neq a$ and also $b \notin R$. Hence, $|C| = |C'| = |R|+1$ and homogeneity provides us with $\pi \in \mathrm{Aut}(\mathbb{Q}, <)$ with $\pi[C] = C'$. Since π is monotone and b is closer to a than any element in R, we necessarily have $\pi(x) = x$ for all $x \in R$ and $\pi(a) = b$.

Note that for general monoids M (i.e. A instead of $\mathbb{Q}$ and M instead of $\mathrm{Aut}(\mathbb{Q}, <)$), homogeneity is a different notion to lock-freeness. Take for instance $A := \Sigma \times \mathbb{A}$ for a finite set Σ, and let M be the finite bijections on A that fix Σ:

$$M := \{f \in \mathfrak{S}_{\mathrm{f}}(\Sigma \times \mathbb{A}) \mid \pi_1(f(\sigma, a)) = \sigma \text{ for all } (\sigma, a) \in A\}$$

Then, M is clearly lock-free: given $(\sigma, a) \in A \backslash R$, just consider the transposition $\ell = ((\sigma, a)\ (\sigma, b))$ for some fresh b. However, M does not fulfil homogeneity. Simply put $C = \{(\sigma, a)\}$, $C' = \{(\sigma, b)\}$ for distinct $a, b \in \mathbb{A}$ and some $\sigma \in \Sigma$; then, there is no $\pi \in M$ with $\pi[C] = C'$. □

Proof of Lemma 4.7

To see that this is indeed equivalent, we have the following chain of equivalences for all $a \in A$ and finite $R \subseteq A$:

$a \notin R$ implies there exists $\ell \in M$ with $\ell \approx_R \mathrm{id}_A$ and $\ell(a) \neq a$.

$$\begin{aligned} &\Leftrightarrow\ a \notin R \implies \exists \ell \in M : \ell \approx_R \mathrm{id}_A \wedge \ell(a) \neq a \\ &\Leftrightarrow\ (\neg\exists \ell \in M : \ell \approx_R \mathrm{id}_A \wedge \ell(a) \neq a) \implies a \in R \\ &\Leftrightarrow\ (\forall \ell \in M : \neg(\ell \approx_R \mathrm{id}_A) \vee \ell(a) = a) \implies a \in R \\ &\Leftrightarrow\ (\forall \ell \in M, \ell \approx_R \mathrm{id}_A\colon \ell(a) = a) \implies a \in R \end{aligned} \qquad □$$

Proof of Lemma 4.8

After Lemma 4.4 it remains to show that $m[S] \subseteq R$ for every finite $R \subseteq A$ that supports $[m]_S$.

Consider an arbitrary $\ell \in M$ with $\ell \approx_R \mathrm{id}_A$. Since R supports $[m]_S$ (Definition 2.8), this implies $\ell \cdot [m]_S = \mathrm{id}_A \cdot [m]_S$. Thus, $[\ell \cdot m]_S = [m]_S$ and so $\ell(m(a)) = m(a)$ for all $a \in S$. So M and R fulfil $\forall \ell \in M, \ell \approx_R \mathrm{id}_A\colon \ell(m(a)) = m(a)$ for all $a \in S$. Hence by (3), we obtain $m(a) \in R$ for all $a \in S$. In other words, $m[S] \subseteq R$. □

Verification of Definition 4.9

For the verification that $M\bullet$ is a functor, consider a supported map $f\colon X \to Y$ and $([m]_{\mathsf{s}(x)}, x) \in M\bullet X$. The map $M\bullet f$ is well-defined, because

$$\mathsf{s}_Y(f(x)) \subseteq \mathsf{s}_X(x) \quad \text{implies} \quad [m]_{\mathsf{s}_X(x)} \subseteq [m]_{\mathsf{s}_Y(f(x))},$$

so if $[m]_{\mathsf{s}(x)} = [m']_{\mathsf{s}(x)}$, then also $[m]_{\mathsf{s}(f(x))} = [m']_{\mathsf{s}(f(x))}$. It is straightforward to see that $M\bullet(-)$ preserves identities and composition. □

Proof of Proposition 4.10

By Lemma 4.4, $M/\mathsf{s}(x)$ is a nominal M-set, and hence the disjoint union

$$M\bullet X = \coprod_{x \in X} M/\mathsf{s}(x)$$

is also a nominal M-set.

For supported map $f\colon X \to Y$, $M\bullet f\colon M\bullet X \to M\bullet Y$ is equivariant because

$$\begin{aligned} \ell \cdot (M\bullet f)([m]_{\mathsf{s}_X(x)}, x) &= \ell \cdot ([m]_{\mathsf{s}_Y(f(x))}, f(x)) \\ &= ([\ell \cdot m]_{\mathsf{s}_Y(f(x))}, f(x)) \\ &= (M\bullet f)([\ell \cdot m]_{\mathsf{s}_X(x)}, x) \end{aligned} \qquad □$$

Proof of Proposition 4.11

The proposed unit η_X is a supported map $\eta_X\colon X \to UFX$ because

$$\begin{aligned} \mathsf{s}_{UFX}(\eta_X(x)) &= \mathsf{s}_{UFX}([\mathrm{id}_A]_{\mathsf{s}(x)}, x) \\ &= \mathrm{supp}([\mathrm{id}_A]_{\mathsf{s}(x)}, x) && \text{(Lemma 3.3)} \\ &\subseteq \mathrm{id}_A[\mathsf{s}_X(x)] = \mathsf{s}_X(x). && \text{(Lemma 4.4)} \end{aligned}$$

For the universal property, consider a supported map $f\colon X \to UY$ (in $\mathsf{Supp}(A)$) for some nominal M-set Y. We need to show that there is a unique $g\colon M\bullet X \to Y$ in $\mathsf{Nom}(M)$ with $g([\mathsf{id}_A]_{\mathsf{s}(x)}, x) = g(\eta_X(x)) = f(x)$:

$$\begin{tikzcd}
U(M\bullet X) \arrow[r, "Ug"] & UY \\
X \arrow[u, "\eta_X"] \arrow[ur, "f"'] &
\end{tikzcd}$$

Define g by

$$g([m]_{\mathsf{s}(x)}, x) := m \cdot f(x)$$

which clearly fulfils $g(\eta_X(x)) = f(x)$. For well-definedness, consider $m \approx_{\mathsf{s}(x)} m'$, and thus $m(a) = m'(a)$ for all $a \in \mathsf{s}_X(x)$. Since M admits least supports, we have

$$\mathsf{supp}_Y(f(x)) = \mathsf{s}_{UY}(f(x)) \subseteq \mathsf{s}_X(x),$$

and so $m(a) = m'(a)$ for all $a \in \mathsf{supp}_Y(f(x))$. Thus, $m \cdot f(x) = m' \cdot f(x)$ (Definition 2.8). Clearly, g is equivariant:

$$\begin{aligned} g(\ell \cdot ([m]_{\mathsf{s}(x)}, x)) &= g([\ell \cdot m]_{\mathsf{s}(x)}, x) = \ell \cdot m \cdot f(x) \\ &= \ell \cdot g([m]_{\mathsf{s}(x)}, x) \qquad \text{for all } \ell \in M. \end{aligned}$$

For uniqueness, consider another equivariant $g'\colon M\bullet X \to Y$ with $g'([\mathsf{id}_A]_{\mathsf{s}(x)}, x) = g'(\eta_X(x)) = f(x)$. By equivariance, we obtain

$$\begin{aligned} g'([m]_{\mathsf{s}(x)}, x) &= g'([m \cdot \mathsf{id}_A]_{\mathsf{s}(x)}, x) = g'(m \cdot ([\mathsf{id}_A]_{\mathsf{s}(x)}, x)) \\ &= m \cdot g'([\mathsf{id}_A]_{\mathsf{s}(x)}, x) \\ &= m \cdot f(x) = g([m]_{\mathsf{s}(x)}, x), \end{aligned}$$

i.e. $g' = g$. □

Proof of Theorem 4.13

We have seen in Proposition 4.11 that $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ is right-adjoint if M admits least supports. For the monadicity, we use Beck's theorem (see e.g. [25, Section VI.7]):

Theorem A.2 (Beck's theorem). *For every right-adjoint functor $U\colon \mathcal{D} \to \mathcal{C}$ the following are equivalent:*

1. *$U\colon \mathcal{D} \to \mathcal{C}$ is monadic.*
2. *The functor $U\colon \mathcal{D} \to \mathcal{C}$ creates split coequalizers: concretely, if $f, g\colon X \to Y$ in $\mathcal{D}$ and morphisms e, r, t in $\mathcal{C}$ make*

$$\begin{tikzcd}
UY \arrow[r, "t"] \arrow[d, "e"'] \arrow[rr, bend left, "\mathsf{id}_{UY}"] & UX \arrow[r, "Uf"] \arrow[d, "Ug"] & UY \arrow[d, "e"] \\
E \arrow[r, "r"] \arrow[rr, bend right, "\mathsf{id}_E"'] & UY \arrow[r, "e"] & E
\end{tikzcd} \quad \text{in } \mathcal{C}$$

commute, then f, g have a coequalizer $e'\colon Y \to E'$ in $\mathcal{D}$ with $Ue' = e$.

Let us verify that $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ reflects split coequalizers: Consider a parallel pair of equivariant maps $f, g\colon X \to Y$ in $\mathsf{Nom}(M)$ and consider supported maps e, r, t such that

$$\begin{tikzcd}
UY \arrow[r, "t"] \arrow[d, "e"'] \arrow[rr, bend left, "\mathsf{id}_{UY}"] & UX \arrow[r, "Uf"] \arrow[d, "Ug"] & UY \arrow[d, "e"] \\
E \arrow[r, "r"] \arrow[rr, bend right, "\mathsf{id}_E"'] & UY \arrow[r, "e"] & E
\end{tikzcd} \quad \text{in } \mathsf{Supp}(A) \tag{5}$$

commutes.

Monoid action $(E, \cdot)$**.** First, define an M-set structure on E using the nominal structure on Y:

$$m \cdot x := e(\underbrace{m \cdot r(x)}_{\text{in } Y}) \qquad \text{for all } m \in M, x \in E$$

This definition implies that

$$m \cdot e(y) = e(m \cdot y) \qquad \text{for all } m \in M, y \in Y, \tag{6}$$

because we have

$$\begin{aligned} m \cdot e(y) &= e(m \cdot r(e(y))) && \text{(Def.)} \\ &= e(m \cdot Ug(t(y))) && (r \cdot e = Ug \cdot t \text{ (5)}) \\ &= e(Ug(\underbrace{m \cdot t(y)}_{\text{in } X})) && (g \text{ equivariant}) \\ &= e(Uf(m \cdot t(y))) && (e \cdot Ug = e \cdot Uf \text{ (5)}) \\ &= e(m \cdot Uf(t(y))) && (f \text{ equivariant}) \\ &= e(m \cdot y). && (Uf \cdot t = \mathsf{id}_{UY} \text{ (5)}) \end{aligned}$$

Since e is surjective, the defined M-set action on E fulfils the required axioms:

$$\begin{aligned} \mathsf{id}_A \cdot e(y) &= e(\mathsf{id}_A \cdot y) = e(y) \\ (k \cdot m) \cdot e(y) &= e((k \cdot m) \cdot y) = e(k \cdot (m \cdot y)) \\ &= k \cdot e(m \cdot y) = k \cdot (m \cdot e(y)). \end{aligned}$$

Also, equation (6) implies that e is an equivariant map. By Lemma 2.14, every element of E is finitely supported. Hence, $(E, \cdot)$ is a nominal M-set, and $e\colon (Y, \cdot) \to (E, \cdot)$ is an equivariant map (i.e. it is in $\mathsf{Nom}(M)$) with $e \cdot f = e \cdot g$. For clarity, we explicitly write (E, s_E) for the originally given supported set and $(E, \cdot)$ to denote the constructed nominal set.

Verification of $U(E, \cdot) = E$**.** We need to verify that $U(E, \cdot) = E$ and in particular $\mathsf{s}_{U(E,\cdot)} = \mathsf{s}_E$. Since M, admits least supports, we have $\mathsf{s}_{U(E,\cdot)} = \mathsf{supp}_{(E,\cdot)}$ (Lemma 3.3). We show the two inclusions of $\mathsf{s}_E(x) = \mathsf{supp}_{(E,\cdot)}(x)$ for all $x \in E$ separately:

- For the proof of $\mathsf{s}_E(x) \supseteq \mathsf{supp}_{(E,\cdot)}(x)$, we verify

$$\begin{aligned} \mathsf{supp}_{(E,\cdot)}(x) &= \mathsf{supp}_{(E,\cdot)}(e(r(x))) \\ &\subseteq \mathsf{supp}_Y(r(x)) && (e \text{ equivariant, Lemma 2.14}) \\ &= \mathsf{s}_{UY}(r(x)) && (\text{Def. } U, \text{ Lemma 3.3}) \\ &\subseteq \mathsf{s}_E(x) && (r \text{ supported map}) \end{aligned}$$

- For the proof of $\mathsf{s}_E(x) \subseteq \mathsf{supp}_{(E,\cdot)}(x)$, consider $a \in \mathsf{s}_E(x)$. We use the contrapositive formulation of M being lock-free (3) for

$$R := \mathsf{supp}_{(E,\cdot)}(x) \cup (\mathsf{s}_E(x) \setminus \{a\}).$$

Hence, we first show that the condition of (3) applies, namely that

$$\forall m \in M\colon \text{ if } m \approx_R \mathsf{id}_A \text{ then } m(a) = a. \tag{$*$}$$

Consider $m \in M$ with $m \approx_R \mathsf{id}_A$. By definition, $R \supseteq \mathsf{supp}_{(E,\cdot)}(x)$ supports x, hence $x = m \cdot x$ and

$$\begin{aligned} a \in \mathsf{s}_E(x) = \mathsf{s}_E(m\cdot x) &= \mathsf{s}_E(e(m \cdot r(x))) && \text{(Def. } m\cdot x\text{)}\\ &\subseteq \mathsf{s}_{UY}(m \cdot r(x)) && (e \text{ supported map})\\ &= \mathsf{supp}_Y(m\cdot r(x)) && (\mathsf{s}_{UY} = \mathsf{supp}_Y \text{ by Lemma 3.3})\\ &\subseteq m[\mathsf{supp}_Y(r(x))] && \text{(Lemma 2.15)}\\ &= m[\mathsf{s}_{UY}(r(x))] \subseteq m[\mathsf{s}_E(x)] && (r \text{ supported map}) \end{aligned}$$

Hence, $a \in m[\mathsf{s}_E(x)]$, so there is some $b \in \mathsf{s}_E(x)$ with $m(b) = a$.
  - If $b \in R$, then $m(b) = \mathsf{id}_A(b)$ since $m \approx_R \mathsf{id}_A$, and so $a = b$.
  - If $b \notin R$, then in particular $b \notin \mathsf{s}_E(x) \setminus \{a\}$. Together with $b \in \mathsf{s}_E(x)$, this implies $b = a$.

In any case, we have deducted $b = a$ and $m(a) = m(b) = a$ the conclusion of $(*)$. Having proven $(*)$, lock-freeness of M provides us with

$$a \in R = \mathsf{supp}_{(E,\cdot)}(x) \cup (\mathsf{s}_E(x) \setminus \{a\}).$$

Since a is clearly not in the right-hand disjunct, it must be in the left disjunct $a \in \mathsf{supp}_{(E,\cdot)}(x)$.

Both inclusions together show that $\mathsf{s}_E = \mathsf{supp}_{(E,\cdot)}$, and so $U(E,\cdot) = E$ and automatically $Ue = e$.

Universal Property of $(E,\cdot)$. It remains to show that $e\colon Y \to (E,\cdot)$ is indeed the coequalizer of f and g in $\mathsf{Nom}(M)$. Since $e\cdot Uf = e\cdot Ug$ in $\mathsf{Supp}(A)$, we also have $e\cdot f = e \cdot g$ in $\mathsf{Nom}(M)$. Given another cocone, i.e. an equivariant map $h\colon Y \to H$ with $h\cdot f = h \cdot g$, let $u\colon E \to UH$ be induced by the coequalizer in $\mathsf{Supp}(A)$, i.e. u is the unique supported map with $u \cdot e = h$. This supported map is equivariant, because for all $m \in M$ and $x \in E$, we straightforwardly have

$$\begin{aligned} m\cdot u(x) &= m\cdot u(e(r(x))) = m \cdot h(r(x)) = h(m\cdot r(x))\\ &= u(e(m\cdot r(x))) \overset{(6)}{=} u(m \cdot e(r(x))) = u(m\cdot x). \end{aligned}$$

Hence, $u\colon (E,\cdot) \to H$ is the desired cocone morphism in $\mathsf{Nom}(M)$. Uniqueness of u is clear because $U\colon \mathsf{Nom}(M) \to \mathsf{Supp}(A)$ is faithful. ◻

Proof of Lemma 5.5

For every supported set X (with s_X), $\mathcal{B}X$ (with $\mathsf{s}_{\mathcal{B}X}$) is a supported set by definition. A supported map $f\colon X \to Y$ is sent by $\mathcal{B}$ to the map

$$\mathcal{B}f\colon \mathcal{B}X \to \mathcal{B}Y \qquad \mathcal{B}f(\lambda.x) = \lambda.f(y).$$

This is again a supported map because

$$\begin{aligned} \mathsf{s}_{\mathcal{B}Y}(\mathcal{B}f(\lambda.x)) &= \mathsf{s}_{\mathcal{B}Y}(\lambda.f(x))\\ &= \{\varrho(k) \mid \varrho(k+1) \in \mathsf{s}_Y(f(x)), k \in \mathbb{N}\} && (4)\\ &\subseteq \{\varrho(k) \mid \varrho(k+1) \in \mathsf{s}_X(x), k\in\mathbb{N}\}\\ &= \mathsf{s}_{\mathcal{B}X}(\lambda.x) && (4) \end{aligned}$$

Since $\mathcal{B}f$ just has f as its underlying map, $\mathcal{B}$ preserves identities and compositions of supported maps. ◻

Proof of Lemma 5.8

In the following, we denote the algebra structure of an Eilenberg-Moore-algebra (C,α) by

$$\mathsf{struct}(C,\alpha) := \alpha.$$

Since the carrier of an Eilenberg-Moore-algebra is given by $U(C,\alpha) = C$, we can write every T-algebra $X := (C,\alpha)$ as a morphism

$$TUX \xrightarrow{\mathsf{struct}(X)} UX$$

without needing to 'unpack' the tuple (C,α). In order to be a strict lifting, the functor $\bar{H}\colon \mathsf{EM}(T) \to \mathsf{EM}(T)$ needs to send a T-algebra $X := (C,\alpha)$ to a T-algebra on HC, so it only remains to define $\mathsf{struct}(\bar{H}X)$ as the composition:

$$\begin{tikzcd} THUX \arrow[r, "T\phi_X"] \arrow[d, dashed, "\mathsf{struct}(\bar{H}X)"', ":="] & TUGX \arrow[d, "\mathsf{struct}(GX)"] \\ HUX & UGX \arrow[l, "\phi_X^{-1}"'] \end{tikzcd}$$

Hence, we have that ϕ_X is a homomorphism of T-algebras by definition and that $\mathsf{struct}(\bar{H}UX)$ is the unique morphism with this property. The unit η of the monad T is preserved, because $\mathsf{struct}(\bar{H}X) \cdot \eta_{HUX} = \mathsf{id}_{HUX}$.

$$\begin{tikzcd} HUX \arrow[r, "\phi_X"] \arrow[d, "\eta_{HUX}"'] \arrow[dr, phantom, "\text{Naturality}"] & UGX \arrow[d, "\eta_{UGX}"] \arrow[dd, bend left=60, "\mathsf{id}_{UGX}"] \\ THUX \arrow[r, "T\phi_X"] \arrow[d, "\mathsf{struct}(\bar{H}X)"'] \arrow[dr, phantom, "\text{Def.}"] & TUGX \arrow[d, "\mathsf{struct}(GX)"] \\ HUX & UGX \arrow[l, "\phi_X^{-1}"'] \end{tikzcd}$$

With the same successive application of the definition of $\mathrm{struct}(\bar{H}X)$, it preserves the multiplication μ of T:

$$\begin{tikzcd}
TTHUX \arrow[rrr, "T\mathrm{struct}(\bar{H}X)"] \arrow[dr, "TT\phi_X"] \arrow[ddd, "\mu_{HUX}"'] & & & THUX \arrow[dl, shift right, "T\phi_X^{-1}"'] \arrow[ddd, "\mathrm{struct}(\bar{H}X)"] \\
\text{Naturality} & TTUGX \arrow[r, "T\mathrm{struct}(GX)"] \arrow[d, "\mu_{UGX}"'] & TUGX \arrow[ur, shift right, "T\phi_X"'] \arrow[d, "\mathrm{struct}(GX)"] & \\
& TUGX \arrow[r, "\mathrm{struct}(GX)"'] \arrow[r, phantom, shift left=4, "\text{Axiom}"] & UGX \arrow[dr, "\phi_X^{-1}"] & \\
THUX \arrow[ur, "T\phi_X"] \arrow[rrr, "\mathrm{struct}(\bar{H}X)"] & & & HUX
\end{tikzcd}$$

For the functoriality of $\bar{H}$, it is easy to see that every T-algebra homomorphism $f\colon X \to Y$ is sent to a homorphism again since G is functorial:

$$\begin{tikzcd}
THUX \arrow[r, "T\phi_X"] \arrow[d, "THUf"'] \arrow[rrr, bend left, "\mathrm{struct}(\bar{H}X)"] \arrow[r, phantom, shift right=6, "\text{Naturality}"] & TUGX \arrow[r, "\mathrm{struct}(GX)"] \arrow[d, "TUGf"] & UGX \arrow[r, "\phi_X^{-1}"] \arrow[d, "UGf"'] \arrow[r, phantom, shift right=6, "\text{Naturality}"] & HUX \arrow[d, "HUf"] \\
THUY \arrow[r, "T\phi_Y"] \arrow[rrr, bend right, "\mathrm{struct}(\bar{H}Y)"'] & TUGY \arrow[r, "\mathrm{struct}(GY)"] & UGY \arrow[r, "\phi_Y^{-1}"] & HUY
\end{tikzcd}$$

Finaly, preservation of identities and composition is immediate. ◻

Proof of Theorem 5.9

We verify that ϕ is indeed a natural isomorphism between $\mathcal{B}U$ and $U[\mathbb{A}]$ by showing that every ϕ_X is a support-reflecting, bijective map (Proposition 3.16), and that it is natural in X:

Support-reflecting: For $\lambda.x \in \mathcal{B}UX$, let $m := \mathrm{maxidx}(x)$:

$$\begin{aligned}
&\mathsf{s}_{U[\mathbb{A}]X}(\phi_X(\lambda.x)) \\
&= \mathsf{s}_{U[\mathbb{A}]X}(\sigma_m^{-1} \cdot \langle\varrho(0)\rangle x) \\
&= \sigma_m^{-1} \cdot \mathsf{supp}_{[\mathbb{A}]X}(\langle\varrho(0)\rangle x) && \text{(Lemma 2.15)} \\
&= \sigma_m^{-1} \cdot (\mathsf{supp}_X(x) \setminus \varrho(0)) \\
&= \sigma_m^{-1} \cdot \{\varrho(k) \mid \varrho(k) \in \mathsf{supp}_X(x), k \neq 0\} \\
&= \{\sigma_m^{-1}(\varrho(k)) \mid \varrho(k) \in \mathsf{supp}_X(x), k \neq 0\} \\
&= \{\varrho(k-1) \mid \varrho(k) \in \mathsf{supp}_X(x), k \neq 0\} && \text{(Def. } \sigma_m\text{)} \\
&= \{\varrho(k) \mid \varrho(k+1) \in \mathsf{supp}_X(x), k \in \mathbb{N}\} \\
&= \{\varrho(k) \mid \varrho(k+1) \in \mathsf{s}_{UX}(x), k \in \mathbb{N}\} \\
&= \mathsf{s}_{\mathcal{B}UX}(\lambda.x) && \text{(by (4))}
\end{aligned}$$

So ϕ_X is indeed support-reflecting.

Surjective: For the surjectivity of $\phi_X\colon \mathcal{B}UX \to U[\mathbb{A}]X$, consider $\langle\varrho(k)\rangle y \in U[\mathbb{A}]X$. With

$$m := \max\{\mathrm{maxidx}(y), k\} + 1$$

and

$$x := \big((\varrho(0)\ \varrho(k+1)) \cdot \sigma_m \cdot y\big),$$

we will that $\phi_X(\lambda.x) = \langle\varrho(k)\rangle y$.

For the definition of ϕ_X, let us first investigate $\mathrm{maxidx}(x)$:

$$\mathsf{supp}(\sigma_m \cdot y) = \{\varrho(\ell+1) \mid \varrho(\ell) \in \mathsf{supp}(y)\}$$

and moreover

$$\mathsf{supp}(x) \subseteq \{\varrho(0)\} \cup \{\varrho(\ell+1) \mid \varrho(\ell) \in \mathsf{supp}(y)\}$$

Hence, $\mathrm{maxidx}(x) \leq \mathrm{maxidx}(y) < m$. Every $\varrho(\ell)$ in the support of $\langle\varrho(0)\rangle x$ is in the range $1 \leq \ell < m$, so

$$\sigma^{-1}_{\mathrm{maxidx}(x)}(\varrho(\ell)) = \varrho(\ell) - 1 = \sigma_m^{-1}(\varrho(\ell))$$

and $\sigma_m^{-1} \approx_S \sigma^{-1}_{\mathrm{maxidx}(x)}$ for $S := \mathsf{supp}(\langle\varrho(0)\rangle x)$ (Definition 2.6). Since S supports $\langle\varrho(0)\rangle x$ (Definition 2.8), these permutations act identically on it, i.e.

$$\sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle x = \sigma_m^{-1} \cdot \langle\varrho(0)\rangle x$$

and we verify:

$$\begin{aligned}
\phi_X(\lambda.x) &= \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle x && \text{(Def. } \phi_X\text{)} \\
&= \sigma_m^{-1} \cdot \langle\varrho(0)\rangle x
\end{aligned}$$

Note that $\varrho(0) \notin \mathsf{supp}(\sigma_m \cdot y)$, so $\varrho(k+1) \notin \mathsf{supp}(x)$, i.e. $\varrho(k+1)$ is fresh for x. Hence,

$$(\varrho(0), x) \sim_\alpha (\varrho(k+1), (\varrho(0)\ \varrho(k+1)) \cdot x)$$

and we conclude:

$$\begin{aligned}
\phi_X(\lambda.x) &= \sigma_m^{-1} \cdot \langle\varrho(0)\rangle x \\
&= \sigma_m^{-1} \cdot \langle\varrho(k+1)\rangle\big((\varrho(0)\ \varrho(k+1)) \cdot x\big) \\
&= \sigma_m^{-1} \cdot \langle\varrho(k+1)\rangle\big(\sigma_m \cdot y\big) && \text{(Def. } x\text{)} \\
&= \langle\sigma_m^{-1}(\varrho(k+1))\rangle\big(\sigma_m^{-1} \cdot \sigma_m \cdot y\big) && \text{(Def. } [\mathbb{A}]\text{)} \\
&= \langle\sigma_m^{-1}(\varrho(k+1))\rangle\big(y\big) \\
&= \langle\varrho(k)\rangle\big(y\big) && (k < m)
\end{aligned}$$

Injective: Consider x, y with

$$\begin{aligned}
\phi_X(\lambda.x) &= \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle x \\
&= \sigma^{-1}_{\mathrm{maxidx}(y)} \cdot \langle\varrho(0)\rangle y = \phi_X(\lambda.y).
\end{aligned}$$

So by assumption already, $\langle\varrho(0)\rangle x$, $\langle\varrho(0)\rangle y \in [\mathbb{A}]X$ are in the same orbit. Since we have proven the support-reflectivity of ϕ_X already, we have

$$\begin{aligned}
\mathsf{s}_{\mathcal{B}UX}(\lambda.x)) &= \mathsf{supp}_{[\mathbb{A}]X}(\phi_X(\lambda.x)) \\
&= \mathsf{supp}_{[\mathbb{A}]X}(\phi_X(\lambda.y)) = \mathsf{s}_{\mathcal{B}UX}(\lambda.y)).
\end{aligned}$$

By the definition of $\mathsf{s}_{\mathcal{B}UX}$, we have

$$\mathsf{s}_{UX}(x) \cup \varrho(0) = \mathsf{s}_{UX}(y) \cup \varrho(0).$$

Since $\langle\varrho(0)\rangle x$, $\langle\varrho(0)\rangle y$ are in the same orbit, $\varrho(0) \in \mathsf{supp}(x)$ iff $\varrho(0) \in \mathsf{supp}(y)$. Hence,

$$\mathsf{s}_{UX}(x) = \mathsf{s}_{UX}(y)$$

and $\mathrm{maxidx}(x) = \mathrm{maxidx}(y)$. So in fact $\langle\varrho(0)\rangle x = \langle\varrho(0)\rangle y$ and by [28, Lemma 4.2] also $x = y$ and $\lambda.x = \lambda.y$ as desired.

Natural: We need to verify that for every equivariant map $f\colon X \to Y$, the diagram

$$\begin{tikzcd} \mathcal{B}UX \arrow[r, "\phi_X"] \arrow[d, "\mathcal{B}Uf"'] & U[\mathbb{A}]X \arrow[d, "U[\mathbb{A}]f"] \\ \mathcal{B}UY \arrow[r, "\phi_Y"] & U[\mathbb{A}]Y \end{tikzcd}$$

commutes. To this end, we verify:

$$\begin{aligned} & U[\mathbb{A}]f(\phi_X(\lambda.x)) \\ &= U[\mathbb{A}]f(\sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle x) \\ &= \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot U[\mathbb{A}]f(\langle\varrho(0)\rangle x) && ([\mathbb{A}]f \text{ equivariant}) \\ &= \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle(f(x)) \end{aligned}$$

Since every $\varrho(\ell) \in \mathrm{supp}(\langle\varrho(0)\rangle(f(x)))$ is in the range

$$1 \le \ell < \mathrm{maxidx}(f(x)) \le \mathrm{maxidx}(x)$$

we have

$$\sigma^{-1}_{\mathrm{maxidx}(x)}(\varrho(\ell)) = \varrho(\ell - 1) = \sigma^{-1}_{\mathrm{maxidx}(f(x))}(\varrho(\ell)).$$

Thus, we can conclude:

$$\begin{aligned} & U[\mathbb{A}]f(\phi_X(\lambda.x)) \\ &= \sigma^{-1}_{\mathrm{maxidx}(x)} \cdot \langle\varrho(0)\rangle(f(x)) \\ &= \sigma^{-1}_{\mathrm{maxidx}(f(x))} \cdot \langle\varrho(0)\rangle(f(x)) \\ &= \phi_Y(\lambda.f(x)) && (\text{Def. } \phi_Y) \\ &= \phi_Y(\mathcal{B}Uf(\lambda.x)) && \square \end{aligned}$$

Details for Construction 6.1

Apply the adjunction of the Eilenberg-Moore category

$$\mathsf{EM}(M\bullet(-)) = \mathsf{Nom}$$

to

$$Q \xrightarrow{c} H(M\bullet Q) \xrightarrow{\phi_Q} UG(M\bullet Q).$$

This yielding the desired equivariant map

$$M\bullet Q \xrightarrow{d} G(M\bullet Q).$$

If Q is finitely presentable (cf. Proposition 3.12), then by [3, Corollary 2.75], $M\bullet Q$ is finitely presentable in Nom, i.e. orbit-finite. □

Proof of Proposition 6.3

- Finite powerset $\mathcal{P}_\mathsf{f}\colon \mathsf{Nom} \to \mathsf{Nom}$ has precisely the same definition in Nom and $\mathsf{Supp}(\mathbb{A})$. In fact, they both lift from $\mathcal{P}_\mathsf{f}$ on Set.
- The uniformly supported powerset functor on Nom is a lifting of the obvious functor $\mathcal{P}'_{\mathsf{ufs}}$ on $\mathsf{Supp}(\mathbb{A})$:

$$\mathcal{P}'_{\mathsf{ufs}}\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$$
$$\mathcal{P}'_{\mathsf{ufs}}(X) = \{S \subseteq X \mid \bigcup_{x\in S} \mathsf{s}_X(x) \text{ is finite}\}$$
$$\mathsf{s}_{\mathcal{P}'_{\mathsf{ufs}}(X)}(S) = \bigcup_{x\in S} \mathsf{s}_X(x)$$

Since $\mathsf{s}_{UX}(x) = \mathrm{supp}_X(x)$ for all nominal sets X, we have that

$$\mathcal{P}'_{\mathsf{ufs}}UX = U\mathcal{P}_{\mathsf{ufs}}X$$

- Every constant functor $C\colon \mathsf{Nom} \to \mathsf{Nom}$, $CX = N$ is the lifting of the constant functor $C'\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$, $C'X = UN$.
- If the family $G_i\colon \mathsf{Nom} \to \mathsf{Nom}$, $i \in I$ are liftings of the functors $H_i\colon \mathsf{Supp}(\mathbb{A}) \to \mathsf{Supp}(\mathbb{A})$, then
  - $\prod_{i\in I} G_i$ is the lifting of $\prod_{i\in I} H_i$, because the right-adjoint $U\colon \mathsf{Nom} \to \mathsf{Supp}(\mathbb{A})$ preserves products:

$$U\prod_{i\in I} G_i \cong \prod_{i\in I} UG_i \cong \prod_{i\in I} H_i U$$

  - Coproducts are given by disjoint union, both in Nom and in $\mathsf{Supp}(\mathbb{A})$. Hence, $U\colon \mathsf{Nom} \to \mathsf{Supp}(\mathbb{A})$ preserves coproducts and so $\coprod_{i\in I} G_i$ is the lifting of $\coprod_{i\in I} H_i$:

$$U\coprod_{i\in I} G_i \cong \coprod_{i\in I} UG_i \cong \coprod_{i\in I} H_i U \qquad \square$$