Separation and Renaming in Nominal Sets

Joshua Moerman
RWTH Aachen University, Germany
joshua@cs.rwth-aachen.de

Jurriaan Rot
University College London, United Kingdom and Radboud University, The Netherlands
jrot@cs.ru.nl

Abstract

Nominal sets provide a foundation for reasoning about names. They are used primarily in syntax with binders, but also, e.g., to model automata over infinite alphabets. In this paper, nominal sets are related to *nominal renaming sets*, which involve arbitrary substitutions rather than permutations, through a categorical adjunction. In particular, the left adjoint relates the separated product of nominal sets to the Cartesian product of nominal renaming sets. Based on these results, we define the new notion of *separated nominal automata*. We show that these automata can be exponentially smaller than classical nominal automata, if the semantics is closed under substitutions.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Nominal sets, Separated product, Adjunction, Automata, Coalgebra

Digital Object Identifier 10.4230/LIPIcs.CSL.2020.31

Related Version An extended version of the paper is available at [19], `https://arxiv.org/abs/1906.00763`.

Funding This work was partially supported by the ERC AdG project 787914 FRAPPANT and a Marie Curie Fellowship (grant code 795119).

Acknowledgements We would like to thank Jamie Gabbay, Gerco van Heerdt, Tom Hirschowitz, Bart Jacobs, and the anonymous reviewers for their useful comments.

1 Introduction

Nominal sets are abstract sets which allow one to reason over sets with names, in terms of permutations and symmetries. Since their introduction in computer science [11], they have been widely used for implementing and reasoning over syntax with binders [22]. Further, nominal techniques have been related to computability theory [4] and automata theory [3], where they provide an elegant means of studying languages over infinite alphabets. This embeds nominal techniques in a broader setting of *symmetry aware computation* [24].

Gabbay, one of the pioneers of nominal techniques described a variation on the theme: *nominal renaming sets* [9, 10]. Nominal renaming sets are equipped with a monoid action of arbitrary (possibly non-injective) substitution of names, in contrast to nominal sets, which only involve a group action of permutations.

In this paper, we further investigate the relationship between nominal renaming sets and nominal sets, and apply the results to nominal automata theory. We start by establishing a categorical adjunction (Section 3):

$$\mathsf{Pm\text{-}Nom} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{Sb\text{-}Nom}\,,$$

where $\mathsf{Pm\text{-}Nom}$ is the usual category of nominal sets and $\mathsf{Sb\text{-}Nom}$ the category of nominal renaming sets. The right adjoint U simply forgets the action of non-injective substitutions.

The functor F was presented by Dowek and Gabbay [6]; it freely extends a nominal set with elements representing the application of such substitutions. For instance, F maps the nominal set $\mathbb{A}^{(*)}$ of all words consisting of distinct atoms to the nominal renaming set $\mathbb{A}^*$ consisting of all words over the atoms.

In fact, the latter equivalence is a consequence of one of the main results of this paper: the left adjoint F maps the *separated product* $X * Y$ of nominal sets to the Cartesian product of nominal renaming sets (Theorem 3.6 & 3.7). The separated product consists of those pairs whose elements have disjoint supports. This is relevant for name abstraction [22], and has also been studied in the setting of presheaf categories, aimed towards separation logic [21]. As a further consequence, under certain conditions, U maps the exponent to the *magic wand* $X \mathrel{-\!\!*} Y$, which is the right adjoint of the separated product.

We apply these connections between nominal sets and renaming sets in the context of automata theory. In terms of expressivity, nominal automata and the more classical register automata are equivalent, but nominal automata have appealing properties that register automata lack, such as unique minimal automata [2]. Unfortunately, moving from register automata to nominal automata can lead to an exponential blow-up in the number of states.[1]

As a motivating example, we consider a language modelling an n-bounded FIFO queue. The input alphabet is given by $\Sigma = \{\mathsf{Put}(a) \mid a \in \mathbb{A}\} \cup \{\mathsf{Pop}\}$, and the output alphabet by $O = \mathbb{A} \cup \{\bot\}$ (here, $\bot$ is a *null* value). The (generalised) language $L_n \colon \Sigma^* \to O$ maps a sequence of queue operations to the resulting top element when starting from the empty queue, or to $\bot$ if this is undefined. The language L_n can be recognised by a nominal (Moore) automaton, but this requires an exponential number of states in n, as the automaton distinguishes internally between all possible equalities among elements in the queue [20].

Based on the observation that L_n is closed under substitutions, we can come up with a *linear* automata-theoretic representation. To this end, we define the new notion of *separated nominal automaton*, where the transition function is only defined for pairs of states and letters with a disjoint support (Section 4). Using the aforementioned categorical framework, we can go back and forth between languages from separated automata and languages which are closed under substitutions. In the FIFO example, the separated automaton obtained from the original nominal automaton has only $n + 2$ states, thus dramatically reducing the number of states. We expect that such a reduction is useful in many applications, such as active learning of register automata [20].

2 Monoid actions and nominal sets

In order to capture both the standard notion of nominal sets [22] and sets with more general renaming actions [10], we start by defining monoid actions.

▶ **Definition 2.1.** *Let $(M, \cdot, 1)$ be a monoid. An M-*set *is a set X together with a function $\cdot \colon M \times X \to X$ such that $1 \cdot x = x$ and $m \cdot (n \cdot x) = (m \cdot n) \cdot x$ for all $m, n \in M$ and $x \in X$. The function $\cdot$ is called an M-*action *and $m \cdot x$ is often written by juxtaposition mx. A function $f \colon X \to Y$ between two M-sets is M-*equivariant *if $m \cdot f(x) = f(m \cdot x)$ for all $m \in M$ and $x \in X$. The class of M-sets together with equivariant maps forms a category M-**Set**.*

[1] Here, "number of states" refers to the number of orbits in the state space.

Let $\mathbb{A} = \{a, b, c, \ldots\}$ be a countable infinite set of *atoms*. The two main instances of M considered in this paper are the monoid

$$\mathsf{Sb} = \{m \colon \mathbb{A} \to \mathbb{A} \mid m(a) \neq a \text{ for finitely many } a\}$$

of all (finite) substitutions (with composition as multiplication), and the monoid

$$\mathsf{Pm} = \{g \in \mathsf{Sb} \mid g \text{ is a bijection}\}$$

of all (finite) permutations. Since Pm is a submonoid of Sb, any Sb-set is also a Pm-set; and any Sb-equivariant map is also Pm-equivariant. This gives rise to a forgetful functor

$$U \colon \mathsf{Sb}\text{-}\mathbf{Set} \to \mathsf{Pm}\text{-}\mathbf{Set}. \tag{1}$$

The set $\mathbb{A}$ is an Sb-set by defining $m \cdot a = m(a)$. Given an M-set X, the set $\mathcal{P}(X)$ of subsets of X is an M-set, with the action defined by direct image.

For a Pm-set X, the *orbit* of an element x is the set $\mathrm{orb}(x) = \{g \cdot x \mid g \in \mathsf{Pm}\}$. We say X is *orbit-finite* if the set $\{\mathrm{orb}(x) \mid x \in X\}$ is finite.

For any monoid M, the category M-**Set** is symmetric monoidal closed. The product of two M-sets is given by the Cartesian product, with the action defined pointwise: $m \cdot (x, y) = (m \cdot x, m \cdot y)$. In M-**Set**, the exponent $X \to^M Y$ is given by the set $\{f \colon M \times X \to Y \mid f \text{ is equivariant}\}$.[2] The action on such an $f \colon M \times X \to Y$ is defined by $(m \cdot f)(n, x) = f(mn, x)$. A good introduction to the construction of the exponent is given by Simmons [28]. If M is a group, a simpler description of the exponent may be given, carried by the set of all functions $f \colon X \to Y$, with the action given by $(g \cdot f)(x) = g \cdot f(g^{-1} \cdot x)$.

2.1 Nominal M-sets

The notion of *nominal* set is usually defined w.r.t. a Pm-action. Here, we use the generalisation to Sb-actions from [10]. Throughout this section, let M denote a submonoid of Sb.

▶ **Definition 2.2.** *Let X be an M-set, and $x \in X$ an element. A set $C \subset \mathbb{A}$ is an* (M)-support *of x if for all $m_1, m_2 \in M$ s.t. $m_1|_C = m_2|_C$ we have $m_1 x = m_2 x$. An M-set X is called* nominal *if every element x has a finite M-support.*

Nominal M-sets and equivariant maps form a full subcategory of M-**Set**, denoted by M-**Nom**. The M-set $\mathbb{A}$ of atoms is nominal. The powerset $\mathcal{P}(X)$ of a nominal set is not nominal in general; the restriction to finitely supported elements is.

If M is a group, then the notion of support can be simplified by using inverses. To see this, first note that, given elements $g_1, g_2 \in M$, $g_1|_C = g_2|_C$ can equivalently be written as $g_2^{-1} g_1|_C = \mathrm{id}\,|_C$. Second, the statement $g_1 x = g_2 x$ can be expressed as $g_2^{-1} g_1 x = x$. Hence, C is a support iff $g|_C = \mathrm{id}_C$ implies $gx = x$ for all g, which is the standard definition for nominal sets over a group [3, 22]. Surprisingly, a similar characterisation also holds for Sb-sets [10]. Moreover, recall that every Sb-set is also a Pm-set; the associated notions of support coincide on nominal Sb-sets, as shown by the following result. In particular, this means that the forgetful functor (1) restricts to $U \colon \mathsf{Sb}\text{-}\mathbf{Nom} \to \mathsf{Pm}\text{-}\mathbf{Nom}$.

▶ **Lemma 2.3.** *[9, Theorem 4.8] Let X be a nominal Sb-set, $x \in X$, and $C \subset \mathbb{A}$. Then C is an Sb-support of x iff it is a Pm-support of x.*

[2] If we write a regular arrow $\to$, then we mean a map in the category. Exponent objects will always be denoted by annotated arrows.

▶ Remark 2.4. It is not true that any Pm-support is an Sb-support. The condition that X is nominal, in the above lemma, is crucial. Let $X = \mathbb{A} \cup \{*\}$ and define the following Sb-action: $m \cdot a = m(a)$ if m is injective, $m \cdot a = *$ if m is non-injective, and $m \cdot * = *$. This is a well-defined Sb-set, but is *not nominal*. Now consider $U(X)$; this is the Pm-set $\mathbb{A} \cup \{*\}$ with the natural action, which is a *nominal* Pm-set! In particular, as a Pm-set each element has a finite support, but as a Sb-set the supports are infinite.

This counterexample is similar to the "exploding nominal sets" in [9], but even worse behaved. We like to call them *nuclear sets*, since an element will collapse when hit by a non-injective map, no matter how far away the non-injectivity occurs.

For $M \in \{\mathsf{Sb}, \mathsf{Pm}\}$, any element $x \in X$ of a nominal M-set X has a *least* finite support (w.r.t. set inclusion). We denote the least finite support of an element $x \in X$ by $\mathsf{supp}(x)$. Note that by Lemma 2.3, the set $\mathsf{supp}(x)$ is independent of whether a nominal Sb-set X is viewed as an Sb-set or a Pm-set. The *dimension* of X is given by $\dim(X) = \max\{|\mathsf{supp}(x)| \mid x \in X\}$, where $|\mathsf{supp}(x)|$ is the cardinality of $\mathsf{supp}(x)$.

We list some basic properties of nominal M-sets, which have known counterparts for the case that M is a group [3], and when $M = \mathsf{Sb}$ [10].

▶ **Lemma 2.5.** *Let X be an M-nominal set. If C supports an element $x \in X$, then $m \cdot C$ supports $m \cdot x$ for all $m \in M$. Moreover, any $g \in \mathsf{Pm}$ preserves least supports: $g \cdot \mathsf{supp}(x) = \mathsf{supp}(gx)$.*

The latter equality does not hold in general for a monoid M. For instance, the "exploding" nominal renaming sets [10] give counterexamples for $M = \mathsf{Sb}$.

▶ **Lemma 2.6.** *Given M-nominal sets X, Y and a map $f\colon X \to Y$, if f is M-equivariant and C supports an element $x \in X$, then C supports $f(x)$.*

The category M-**Nom** is symmetric monoidal closed, with the product inherited from M-**Set**, thus simply given by Cartesian product. The exponent is given by the restriction of the exponent $X \to^M Y$ in M-**Set** to the set of finitely supported functions, denoted by $X \to^M_{\mathrm{fs}} Y$. This is similar to the exponents of nominal sets with 01-substitutions from [23].

▶ Remark 2.7. In [10] a different presentation of the exponent in M-**Nom** is given, based on a certain extension of partial functions. We prefer the previous characterisation, as it is derived in a straightforward way from the exponent in M-**Set**.

2.2 Separated product

▶ **Definition 2.8.** *Let X and Y be* Pm*-nominal sets. Two elements $x \in X, y \in Y$ are called* separated*, denoted by $x \# y$, if there are disjoint sets $C_1, C_2 \subset \mathbb{A}$ such that C_1 supports x and C_2 supports y. The* separated product *of* Pm*-nominal sets X and Y is defined as*

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

We extend the separated product to the *separated power*, defined by $X^{(0)} = 1$ and $X^{(n+1)} = X^{(n)} * X$, and the *set of separated words* $X^{(*)} = \bigcup_i X^{(i)}$. The separated product is an equivariant subset $X * Y \subseteq X \times Y$. Consequently, we have equivariant projection maps $X * Y \to X$ and $X * Y \to Y$.

▶ **Example 2.9.** Two finite sets $C, D \in \mathcal{P}(\mathbb{A})$ are separated precisely when they are disjoint. An important example is the set $\mathbb{A}^{(*)}$ of separated words over the atoms: it consists of those words where all letters are distinct.

The separated product gives rise to another symmetric closed monoidal structure on Pm-**Nom**, with 1 as unit, and the exponential object given by *magic wand* $X \mathrel{-\!\!*} Y$. An explicit characterisation of $X \mathrel{-\!\!*} Y$ is not needed in the remainder of this paper, but for a complete presentation we briefly recall the description from [26] (see also [22] and [5]). First, define a Pm-action on the set of partial functions $f\colon X \rightharpoonup Y$ by $(g \cdot f)(x) = g \cdot f(g^{-1} \cdot x)$ if $f(g^{-1} \cdot x)$ is defined. Now, such a partial function $f\colon X \rightharpoonup Y$ is called *separating* if f is finitely supported, $f(x)$ is defined iff $f \# x$, and $\mathsf{supp}(f) = \bigcup_{x \in \mathrm{dom}(f)} \mathsf{supp}(f(x)) \setminus \mathsf{supp}(x)$. Finally, $X \mathrel{-\!\!*} Y = \{f\colon X \rightharpoonup Y \mid f \text{ is separating}\}$. See [26] for a proof and explanation.

▶ Remark 2.10. The special case $\mathbb{A} \mathrel{-\!\!*} Y$ coincides with $[\mathbb{A}]Y$, the set of *name abstractions* [22]. The latter is generalised to $[X]Y$ in [8]. In [5] it is shown that the coincidence $[X]Y \cong (X \mathrel{-\!\!*} Y)$ only holds under strong assumptions (including that X is single-orbit).

▶ Remark 2.11. An analogue of the separated product does not seem to exist for nominal Sb-sets. For instance, consider the set $\mathbb{A} \times \mathbb{A}$. As a Pm-set, it has four equivariant subsets: $\emptyset, \{(a,a) \mid a \in \mathbb{A}\}$, $\mathbb{A} * \mathbb{A}$, and $\mathbb{A} \times \mathbb{A}$. However, the set $\mathbb{A} * \mathbb{A}$ is not an equivariant subset when considering $\mathbb{A} \times \mathbb{A}$ as an Sb-set.

3 A monoidal construction from Pm-sets to Sb-sets

In this section, we provide a free construction, extending nominal Pm-sets to nominal Sb-sets. We use this as a basis to relate the separated product and exponent (in Pm-**Nom**) to the product and exponent in Sb-**Nom**. The main results are:

1. Theorem 3.6: the forgetful functor $U\colon$ Sb-**Nom** $\to$ Pm-**Nom** has a left adjoint F;
2. Theorem 3.7: this F is monoidal: it maps separated products to products;
3. Theorem 3.13 and Corollary 3.14: U maps the exponent object in Sb-**Nom** to the right adjoint $\mathrel{-\!\!*}$ of the separated product, if the domain has dimension smaller or equal to 1.

Together, these results form the categorical infrastructure to relate nominal languages to separated languages and automata in Section 4.

▶ **Definition 3.1** (From [6]). *Given a* Pm*-nominal set X, we define a nominal* Sb*-set $F(X)$ as follows. Define the set*

$$F(X) = \{(m, x) \mid m \in \mathsf{Sb}, x \in X\}/_\sim,$$

where $\sim$ is the least equivalence relation containing:

$$(m, gx) \sim (mg, x), \tag{2}$$

$$(m, x) \sim (m', x) \quad \text{if } m|_C = m'|_C \text{ for a Pm-support } C \text{ of } x, \tag{3}$$

for all $x \in X$, $m, m' \in \mathsf{Sb}$ and $g \in \mathsf{Pm}$. Note that $mg = m \circ g$, i.e., simply the monoid operation of Sb*. The equivalence class of a pair (m, x) is denoted by $[m, x]$. We define an* Sb*-action on $F(X)$ as $n \cdot [m, x] = [nm, x]$.*

Well-definedness is proved as part of Proposition 3.5 below. Informally, an equivalence class $[m, x] \in F(X)$ behaves "as if m acted on x." The first equation (2) ensures compatibility with the Pm-action on x, and the second equation (3) ensures that $[m, x]$ only depends the relevant part of m.

▶ **Example 3.2.** Here are a few examples of the application of F. We do not give direct proofs, but the first two will be treated more systematically later in this section (see Corollary 3.12). For the third, note that $\mathbb{A} \times \mathbb{A}$ consist of two orbits, $\mathbb{A} * \mathbb{A}$ and the diagonal $\{(a,a) \mid a \in \mathbb{A}\}$.

- $F(\mathbb{A}) \cong \mathbb{A}$.
- $F(\mathbb{A}^{(*)}) \cong \mathbb{A}^*$.
- $F(\mathbb{A} \times \mathbb{A}) \cong \mathbb{A}^2 + \mathbb{A}$.

The first example is also given in [6], where it is additionally shown that F does not preserve products. As we will see in Section 3.1, F does preserve a different monoidal structure, namely the separated product. The following characterisation of $\sim$ is useful in proofs. This lemma is proven in [19].

▶ **Lemma 3.3.** *We have $(m_1, x_1) \sim (m_2, x_2)$ iff there is a permutation $g \in \mathsf{Pm}$ such that $gx_1 = x_2$ and $m_1|_C = m_2 g|_C$, for C some Pm-support of x_1.*

▶ Remark 3.4. The first relation (2) in Definition 3.1 comes from the construction of "extension of scalars" in commutative algebra [1]. In that context, one has a ring homomorphism $f\colon A \to B$ and an A-module M and wishes to obtain a B-module. This is constructed by the tensor product $B \otimes_A M$ and it is here that the relation $(b, am) \sim (ba, m)$ is used (B is a right A-module via f).

In [6] it is stated that F is a functor, and a proof outline for well-definedness on arrows is given. Here we give a full proof, including well-definedness on objects.

▶ **Proposition 3.5.** *The construction F in Definition 3.1 extends to a functor*

$$F\colon \mathsf{Pm}\text{-}\mathbf{Nom} \to \mathsf{Sb}\text{-}\mathbf{Nom}\,,$$

defined on an equivariant map $f\colon X \to Y$ by $F(f)([m, x]) = [m, f(x)] \in F(Y)$.

Proof. We first prove well-definedness and then the functoriality.

$F(X)$ **is an** Sb**-set.** To this end we check that the Sb-action is well-defined. Let $[m_1, x_1] = [m_2, x_2] \in F(X)$ and let $m \in \mathsf{Sb}$. By Lemma 3.3, there is some permutation g such that $gx_1 = x_2$ and $m_1|_C = m_2 g|_C$ for some support C of x_1. By post-composition with m we get $mm_1|_C = mm_2 g|_C$, which means (again by the lemma) that $[mm_1, x_1] = [mm_2, x_2]$. Thus $m[m_1, x_1] = m[m_2, x_2]$, which concludes well-definedness.

For associativity and unitality of the Sb-action, we note that it is directly defined by left multiplication of Sb which is associative and unital. This concludes that $F(X)$ is an Sb-set.

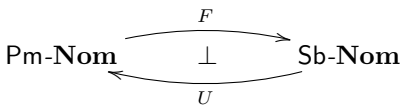
$F(X)$ **is a nominal** Sb **set.** Given an element $[m, x] \in F(X)$ and a Pm-support C of x, we will prove that $m \cdot C$ is an Sb-support for $[m, x]$. Suppose that we have $m_1, m_2 \in \mathsf{Sb}$ such that $m_1|_{m \cdot C} = m_2|_{m \cdot C}$. By pre-composition with m we get $m_1 m|_C = m_2 m|_C$ and this leads us to conclude $[m_1 m, x] = [m_2 m, x]$. So $m_1[m, x] = m_2[m, x]$ as required.

Functoriality. Let $f\colon X \to Y$ be a Pm-equivariant map. To see that $F(f)$ is well-defined consider $[m_1, x_1] = [m_2, x_2]$. By Lemma 3.3, there is a permutation g such that $gx_1 = x_2$ and $m_1|_C = m_2 g|_C$ for some support C of x_1. Applying $F(f)$ gives on one hand $[m_1, f(x_1)]$ and on the other hand $[m_2, f(x_2)] = [m_2, f(gx_1)] = [m_2, gf(x_1)] = [m_2 g, f(x_1)]$ (we used equivariance in the second step). Since $m_1|_C = m_2 g|_C$ and f preserves supports we have $[m_2 g, f(x_1)] = [m_1, f(x_1)]$.

For Sb-equivariance we consider both $n \cdot F(f)([m, x]) = n[m, f(x)] = [nm, f(x)]$ and $F(f)(n \cdot [m, x]) = F(f)([nm, x]) = [nm, f(x)]$. This shows $nF(f)([m, x]) = F(f)(n[m, x])$ and concludes that we have a map $F(f)\colon F(X) \to F(Y)$.

Preservation of the identity function and composition follows from the definition. ◀

▶ **Theorem 3.6.** *The functor F is left adjoint to U:*



Proof. We define, for every nominal set X, a map $\eta_X \colon X \to UF(X)$ with the necessary universal property: for every Pm-equivariant $f \colon X \to U(Y)$ there is a unique Sb-equivariant map $f^\sharp \colon FX \to Y$ such that $U(f^\sharp) \circ \eta_X = f$. Define $\eta_X(x) = [\mathrm{id}, x]$. This is equivariant: $g \cdot \eta_X(x) = g[\mathrm{id}, x] = [g, x] = [\mathrm{id}, gx] = \eta_X(gx)$. Now, for $f \colon X \to U(Y)$, define $f^\sharp([m, x]) = m \cdot f(x)$ for $x \in X$ and $m \in \mathsf{Sb}$. Then $U(f^\sharp) \circ \eta_X(x) = f^\sharp([\mathrm{id}, x]) = \mathrm{id} \cdot f(x) = f(x)$.

For well-definedness of $f^\sharp$, consider $[m_1, x_1] = [m_2, x_2]$ (we have to prove that $m_1 \cdot f(x_1) = m_2 \cdot f(x_2)$). By Lemma 3.3, there is a $g \in \mathsf{Pm}$ such that $gx_1 = x_2$ and $m_2 g|_C = m_1|_C$ for a Pm-support C of x_1. Now C is also a Pm-support for $f(x)$ and hence it is an Sb-support of $f(x)$ (Lemma 2.3). Thus $m_2 \cdot f(x_2) = m_2 \cdot f(gx_1) = m_2 g \cdot f(x_1) = m_1 \cdot f(x_1)$ (we use Pm-equivariance in the one but last step and Sb-support in the last step). For Sb-equivariance, we compute $n \cdot f^\sharp([m, x]) = nm \cdot f(x) = f^\sharp([nm, x]) = f^\sharp(n[m, x])$. For uniqueness, suppose $h \colon FX \to Y$ is such that $U(h) \circ \eta_X = f$, i.e., $h([\mathrm{id}, x]) = f(x)$. Then $h([m, x]) = h(m[\mathrm{id}, x]) = m \cdot h([\mathrm{id}, x]) = m \cdot f(x) = m \cdot f^\sharp([\mathrm{id}, x]) = f^\sharp(m \cdot [\mathrm{id}, x]) = f^\sharp([m, x])$. ◀

The counit $\epsilon \colon FU(Y) \to Y$ is given by $\epsilon([m, x]) = m \cdot x$. For the inverse of $-^\sharp$, let $g \colon F(X) \to Y$ be an Sb-equivariant map; then $g^\flat \colon X \to U(Y)$ is given by $g^\flat(x) = g([\mathrm{id}, x])$. Note that the unit η is a Pm-equivariant map, hence it preserves supports (i.e., any support of x also supports $[\mathrm{id}, x]$). This also means that if C is a support of x, then $m \cdot C$ is a support of $[m, x]$ (by Lemma 2.5).

3.1 On (separated) products

The functor F not only preserves coproducts, being a left adjoint, but it also maps the separated product to products:

▶ **Theorem 3.7.** *The functor F is strong monoidal, from the monoidal category $(\mathsf{Pm}\text{-}\mathbf{Nom}, *, 1)$ to $(\mathsf{Sb}\text{-}\mathbf{Nom}, \times, 1)$. In particular, the map p given by*

$$p = \langle F(\pi_1), F(\pi_2) \rangle \colon F(X * Y) \to F(X) \times F(Y)$$

is an isomorphism, natural in X and Y.

Proof. We prove that p is an isomorphism. It suffices to show that p is injective and surjective. Note that $p([m, (x, y)]) = ([m, x], [m, y])$.

Surjectivity. Let $([m_1, x], [m_2, y])$ be an element of $F(X) \times F(Y)$. We take an element $y' \in Y$ such that $y' \# x$ and $y' = gy$ for some $g \in \mathsf{Pm}$. Now we have an element $(x, y') \in X * Y$. By Lemma 2.5, we have $\mathsf{supp}(y') = g\,\mathsf{supp}(y)$. Define the map

$$m(a) = \begin{cases} m_1(a) & \text{if } a \in \mathsf{supp}(x) \\ m_2(g^{-1}(a)) & \text{if } a \in \mathsf{supp}(y') \\ a & \text{otherwise.} \end{cases}$$

(Observe that $\mathsf{supp}(x) \# \mathsf{supp}(y')$, so the cases are not overlapping.) The map m is an element of Sb. Now consider the element $z = [m, (x, y')] \in F(X * Y)$. Applying p to z gives the element $([m, x], [m, y'])$. First, we note that $[m, x] = [m_1, x]$ by the definition of m. Second, we show that $[m, y'] = [m_2, y]$. Observe that $mg|_{\mathsf{supp}(y)} = m_2|_{\mathsf{supp}(y)}$ by definition of m. Since $\mathsf{supp}(y)$ is a support of y, we have $[mg, y] = [m_2, y]$, and since $[mg, y] = [m, gy] = [m, y']$ we are done. Hence $p([m, (x, y')]) = ([m, x], [m, y']) = ([m_1, x], [m_2, y])$, so p is surjective.

Injectivity. Let $[m_1, (x_1, y_1)]$ and $[m_2, (x_2, y_2)]$ be two elements. Suppose that they are mapped to the same element, i.e., $[m_1, x_1] = [m_2, x_2]$ and $[m_1, y_1] = [m_2, y_2]$. Then there are permutations g_x, g_y such that $x_2 = g_x x_1$ and $y_2 = g_y y_1$. Moreover, let $C = \mathsf{supp}(x_1)$ and

$D = \mathsf{supp}(y_1)$; then we have $m_1|_C = m_2 g_x|_C$ and $m_1|_D = m_2 g_y|_D$. In order to show the two original elements are equal, we have to provide a single permutation g. Define for, $z \in C \cup D$,

$$g_0(z) = \begin{cases} g_x(z) & \text{if } z \in C \\ g_y(z) & \text{if } z \in D. \end{cases}$$

(Again, C and D are disjoint.) The function g_0 is injective since the least supports of x_2 and y_2 are disjoint. Hence g_0 defines a local isomorphism from $C \cup D$ to $g_0(C \cup D)$. By homogeneity [22], the map g_0 extends to a permutation $g \in \mathsf{Pm}$ with $g(z) = g_x(z)$ for $z \in C$ and $g(z) = g_y(z)$ for $z \in D$. In particular we get $(x_2, y_2) = g(x_1, y_1)$. We also obtain $m_1|_{C \cup D} = m_2 g|_{C \cup D}$. Thus $[m_1, (x_1, y_1)] = [m_2, (x_2, y_2)]$, and so the map p is injective.

Unit and coherence. To show that F preserves the unit, we note that $[m, 1] = [m', 1]$ for every $m, m' \in \mathsf{Sb}$, as the empty set supports 1 and so $m|_\emptyset = m'|_\emptyset$ vacuously holds. We conclude $F(1)$ is a singleton. ◀

Since F also preserves coproducts (being a left adjoint), we obtain that F maps the set of separated words to the set of all words.

▶ **Corollary 3.8.** *For any* Pm*-nominal set* X*, we have* $F(X^{(*)}) \cong (FX)^*$*.*

As we will show below, the functor F preserves the set $\mathbb{A}$ of atoms. This is an instance of a more general result about preservation of one-dimensional objects.

▶ **Proposition 3.9.** *The functors* F *and* U *are equivalences on* ≤ 1*-dimensional objects. Concretely, for* $X \in \mathsf{Pm}$-**Nom** *and* $Y \in \mathsf{Sb}$-**Nom***:*

1. *If* $\dim(X) \leq 1$*, then the unit* $\eta \colon X \to UF(X)$ *is an isomorphism.*
2. *If* $\dim(Y) \leq 1$*, then the co-unit* $\epsilon \colon FU(Y) \to Y$ *is an isomorphism.*

In the proof, we will use the following property of Sb-sets with dimension ≤ 1.

▶ **Lemma 3.10.** *Let* Y *be a nominal* Sb*-set. If an element* $y \in Y$ *is supported by a singleton set (or even the empty set), then*

$$\{my \mid m \in \mathsf{Sb}\} = \{gy \mid g \in \mathsf{Pm}\}\,.$$

Proof. Let $y \in Y$ be supported by $\{a\}$ and let $m \in \mathsf{Sb}$. Now consider $b = m(a)$ and the bijection $g = (a\,b)$. Now $m|_{\{a\}} = g|_{\{a\}}$, meaning that $my = gy$. So the set $\{my \mid m \in \mathsf{Sb}\}$ is contained in $\{gy \mid g \in \mathsf{Pm}\}$. The inclusion the other way is trivial, which means $\{my \mid m \in \mathsf{Sb}\} = \{gy \mid g \in \mathsf{Pm}\}$. ◀

Proof of Proposition 3.9. It is easy to see that $\eta \colon x \mapsto [\mathrm{id}, x]$ is injective. Now to see that η is surjective, let $[m, x] \in UF(X)$ and consider a support $\{a\}$ of x (this is a singleton or empty since $\dim(X) \leq 1$). Let $b = m(a)$ and consider the swap $g = (a\,b)$. Now $[m, x] = [mg^{-1}, gx]$ and note that $\{b\}$ supports gx and $mg^{-1}|_{\{b\}} = \mathrm{id}\,|_{\{b\}}$. We conclude with $[mg^{-1}, gx] = [\mathrm{id}, gx]$, which implies that gx is the preimage of $[m, x]$. Hence η is an isomorphism.

To see that $\epsilon \colon [m, y] \mapsto my$ is surjective, just consider $m = \mathrm{id}$. To see that ϵ is injective, let $[m, y], [m', y'] \in FU(Y)$ be two elements such that $my = m'y'$. Then by using Lemma 3.10 we find $g, g' \in \mathsf{Pm}$ such that $gy = my = m'y' = g'y'$. This means that y and y' are in the same orbit (of $U(Y)$) and have the same dimension. Case 1: $\mathsf{supp}(y) = \mathsf{supp}(y') = \emptyset$, then $[m, y] = [\mathrm{id}, y] = [\mathrm{id}, y'] = [m', y']$. Case 2: $\mathsf{supp}(y) = \{a\}$ and $\mathsf{supp}(y') = \{b\}$, then $\mathsf{supp}(gy) = \{g(a)\}$ (Lemma 2.5). In particular we have that m and g map a to $c = g(a)$, likewise m' and g' map b to c. Now $[m, y] = [m, g^{-1}g'y'] = [mg^{-1}g', y'] = [m', y']$, where we used $mg^{-1}g(b) = c = m'(b)$ in the last step. Thus ϵ is injective and hence an isomorphism. ◀

By Proposition 3.9, we may consider the set $\mathbb{A}$ as both Sb-set and Pm-set (abusing notation). And we get an isomorphism $F(\mathbb{A}) \cong \mathbb{A}$ of nominal Sb-sets. To appreciate the above results, we give a concrete characterisation of one-dimensional nominal sets:

▶ **Lemma 3.11.** *Let X be a nominal M-set, for $M \in \{\mathsf{Sb}, \mathsf{Pm}\}$. Then $\dim(X) \leq 1$ iff there exist discrete*[3] *sets Y and I such that $X \cong Y + \coprod_I \mathbb{A}$.*

In particular, the one-dimensional objects include the alphabets used for *data words*, consisting of a product $S \times \mathbb{A}$ of a discrete set S of action labels and the set of atoms. These alphabets are very common in the study of register automata (see, e.g., [13]).

By the above and Theorem 3.7, F maps separated powers of $\mathbb{A}$ to powers, and the set of separated words over $\mathbb{A}$ to the Sb-set of words over $\mathbb{A}$.

▶ **Corollary 3.12.** *We have $F(\mathbb{A}^{(n)}) \cong \mathbb{A}^n$ and $F(\mathbb{A}^{(*)}) \cong \mathbb{A}^*$.*

3.2 On exponents

We have described how F and U interact with (separated) products. Next, we establish a relationship between the magic wand $(-\!\!*)$ and the exponent of nominal Sb-sets $(\to^{\mathsf{Sb}}_{\mathrm{fs}})$. These results on exponents will be useful in Section 4.1, where we discuss automata using coalgebras.

▶ **Theorem 3.13.** *The sets $X \mathrel{-\!\!*} U(Y)$ and $U(F(X) \to^{\mathsf{Sb}}_{\mathrm{fs}} Y)$ are naturally isomorphic as nominal Pm-sets.*

Proof. We have the composite adjunctions

$$F \circ (X * -) \dashv (X \mathrel{-\!\!*} -) \circ U \quad \text{and} \quad (FX \times -) \circ F \dashv U \circ (FX \to^{\mathsf{Sb}}_{\mathrm{fs}} -)\,.$$

Theorem 3.7 gives a natural isomorphism between the left adjoints. Hence, the right adjoints are also isomorphic, which is the desired result. ◀

Note that this theorem gives an alternative characterisation of the magic wand in terms of the exponent in Sb-$\mathbf{Nom}$, if the codomain is $U(Y)$. Moreover, for a 1-dimensional object X in Sb-$\mathbf{Nom}$, we obtain the following special case of the theorem (using the co-unit isomorphism from Proposition 3.9):

▶ **Corollary 3.14.** *Let X, Y be nominal Sb-sets. For 1-dimensional X, the nominal Pm-set $U(X) \mathrel{-\!\!*} U(Y)$ is naturally isomorphic to $U(X \to^{\mathsf{Sb}}_{\mathrm{fs}} Y)$.*

▶ Remark 3.15. The set $\mathbb{A} \mathrel{-\!\!*} U(X)$ coincides with the atom abstraction $[\mathbb{A}]UX$ (Remark 2.10). Hence, as a special case of Corollary 3.14, we recover [10, Theorem 34], which states a bijective correspondence between $[\mathbb{A}]UX$ and $U(\mathbb{A} \to^{\mathsf{Sb}}_{\mathrm{fs}} X)$.

4 Nominal and separated automata

In this section, we study nominal (Moore) automata, which recognise languages over infinite alphabets. After recalling the basic definitions, we introduce a new variant of automata based on the separating product, which we call *separated nominal automata*. These automata

[3] Any set Z can be equipped with a trivial action $m \cdot x = x$, which makes Z a nominal set. Such sets are called discrete.

represent nominal languages which are Sb-equivariant, essentially meaning they are closed under substitution. Our main result is that, if a "classical" nominal automaton (over Pm) represents a language L which is Sb-equivariant, then L can also be represented by a separated nominal automaton. The latter can be exponentially smaller (in number of orbits) than the original automaton, as we show in a concrete example.

▶ Remark 4.1. We will work with a general output set O instead of just acceptance. The reason for this is that Sb-equivariant functions $L\colon \mathbb{A}^{(*)} \to 2$ are not very interesting: they are defined purely by the length of the input. By using more general output O, we may still capture interesting behaviour, e.g., the language in Example 4.3.

▶ **Definition 4.2.** *Let Σ, O be Pm-sets, called input/output alphabet respectively.*

- *A* (Pm)-nominal language *is an equivariant map of the form $L\colon \Sigma^* \to O$.*
- *A* nominal (Moore) automaton *$\mathcal{A} = (Q, \delta, o, q_0)$ consists of a nominal set of states Q, an equivariant transition function $\delta\colon Q \times \Sigma \to Q$, an equivariant output function $o\colon Q \to O$, and an initial state $q_0 \in Q$ with an empty support.*
- *The* language semantics *is the map $l\colon Q \times \Sigma^* \to O$, defined inductively by*

$$l(x, \varepsilon) = o(x)\,, \qquad l(x, aw) = l(\delta(x, a), w)$$

for all $x \in Q$, $a \in \Sigma$ and $w \in \Sigma^*$.

- *For $l^\flat\colon Q \to (\Sigma^* \to^{\mathsf{Pm}}_{\mathrm{fs}} O)$ the transpose of l, we have that $l^\flat(q_0)\colon \Sigma^* \to O$ is equivariant; this is called the* language accepted by *$\mathcal{A}$.*

Note that the language accepted by an automaton can equivalently be characterised by considering paths through the automaton from the initial state.

If the state space Q and the alphabets Σ, O are orbit finite, this allows us to run algorithms (reachability, minimisation, etc.) on such automata [3], but there is no need to assume this for now. For an automaton $\mathcal{A} = (Q, \delta, o, q_0)$, we define the set of *reachable states* as the least set $R(\mathcal{A}) \subseteq Q$ such that $q_0 \in R(\mathcal{A})$ and for all $x \in R(\mathcal{A})$ and $a \in \Sigma$, $\delta(x, a) \in R(\mathcal{A})$.

▶ **Example 4.3.** We model a bounded FIFO queue of size n as a nominal Moore automaton, explicitly handling the data in the automaton structure.[4] The input alphabet Σ and output alphabet O are as follows:

$$\Sigma = \{\mathsf{Put}(a) \mid a \in \mathbb{A}\} \cup \{\mathsf{Pop}\}, \qquad O = \mathbb{A} \cup \{\bot\}.$$

The input alphabet encodes two actions: putting a new value on the queue and popping a value. The output is either a value (the front of the queue) or $\bot$ if the queue is empty. A queue of size n is modelled by the automaton (Q, δ, o, q_0) defined as follows.

$$Q = \mathbb{A}^{\leq n} \cup \{\bot\} \qquad q_0 = \varepsilon \qquad o(a_1 \dots a_k) = \begin{cases} a_1 & \text{if } k \geq 1 \\ \bot & \text{otherwise} \end{cases}$$

$$\delta(a_1 \dots a_k, \mathsf{Put}(b)) = \begin{cases} a_1 \dots a_k b & \text{if } k < n \\ \bot & \text{otherwise} \end{cases} \qquad \delta(\bot, x) = \bot$$

$$\delta(a_1 \dots a_k, \mathsf{Pop}) = \begin{cases} a_2 \dots a_k & \text{if } k > 0 \\ \bot & \text{otherwise} \end{cases}$$

[4] We use a reactive version of the queue data structure, which slightly differs from the versions in [20, 13].

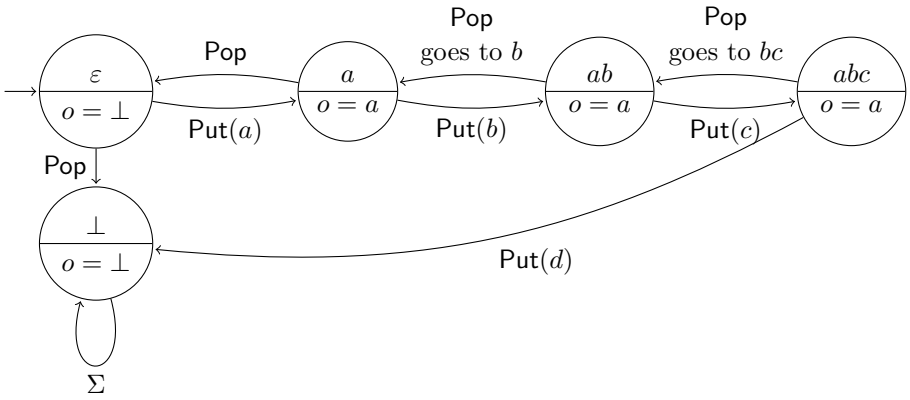


Figure 1 The FIFO automaton from Example 4.3 with $n = 3$. The right-most state consists of *five* orbits as we can take a, b, c distinct, all the same, or two of them equal in three different ways. Consequently, the complete state space has ten orbits. The output of each state is denoted in the lower part.

The automaton is depicted in Figure 1 for the case $n = 3$. The language accepted by this automaton assigns to a word w the first element of the queue after executing the instructions in w from left to right, and $\bot$ if the input is ill-behaved, i.e., Pop is applied to an empty queue or Put(a) to a full queue.

▶ **Definition 4.4.** *Let Σ, O be* Pm*-sets. A* separated language *is an equivariant map of the form $\Sigma^{(*)} \to O$. A* separated automaton *$\mathcal{A} = (Q, \delta, o, q_0)$ consists of Q, o and q_0 defined as in a nominal automaton, and an equivariant transition function $\delta \colon Q * \Sigma \to Q$.*

The separated language semantics *of such an automaton $\mathcal{A}$ is given by the function $s \colon Q * \Sigma^{(*)} \to O$, defined inductively by*

$$s(x, \varepsilon) = o(x)\,, \qquad s(x, aw) = s(\delta(x, a), w)$$

for all $x \in Q$, $a \in \Sigma$ and $w \in \Sigma^{(*)}$ such that $x \# aw$ and $a \# w$.

Let $s^\flat \colon Q \to (\Sigma^{(*)} \nrightarrow O)$ be the transpose of s. Then $s^\flat(q_0) \colon \Sigma^{(*)} \to O$ corresponds to a separated language; this is called the separated language accepted by *$\mathcal{A}$.*

By definition of the separated product, the transition function is only defined on a state x and letter $a \in \Sigma$ if $x \# a$. In Example 4.10 below, we describe the bounded FIFO as a separated automaton, and describe its accepted language.

First, we show how the language semantics of separated nominal automata extends to a language over *all* words, provided that both the input alphabet Σ and the output alphabet O are Sb-sets.

▶ **Definition 4.5.** *Let Σ and O be nominal* Sb*-sets. An* Sb*-equivariant function $L \colon \Sigma^* \to O$ is called an* Sb-language*.*

Notice the difference between an Sb-language $L \colon \Sigma^* \to O$ and a Pm-language $L' \colon (U\Sigma)^* \to U(O)$. They are both functions from Σ^* to O, but the latter is only Pm-equivariant, while the former satisfies the stronger property of Sb-equivariance. Languages over separated words, and Sb-languages, are connected as follows.

▶ **Proposition 4.6.** *Suppose Σ, O are both nominal* Sb*-sets, and suppose* $\dim(\Sigma) \leq 1$*. There is a one-to-one correspondence*

$$\frac{S\colon (U\Sigma)^{(*)} \to UO \quad \text{Pm-}\textit{equivariant}}{\overline{S}\colon \Sigma^* \to O \quad \text{Sb-}\textit{equivariant}}$$

between separated languages and Sb*-nominal languages. From $\overline{S}$ to S, this is given by application of the forgetful functor and restricting to the subset of separated words.*

For the converse direction, given $w = a_1 \dots a_n \in \Sigma^*$, let $b_1, \dots, b_n \in \Sigma$ such that $w \# b_i$ for all i, and $b_i \# b_j$ for all i, j with $i \neq j$. Define $m \in \mathsf{Sb}$ by

$$m(a) = \begin{cases} a_i & \text{if } a = b_i \text{ for some } i \\ a & \text{otherwise} \end{cases}$$

Then $\overline{S}(a_1 a_2 a_3 \cdots a_n) = m \cdot S(b_1 b_2 b_3 \cdots b_n)$.

Proof. There is the following chain of one-to-one correspondences, from the results of the previous section:

$$\begin{array}{ll} \dfrac{(U\Sigma)^{(*)} \to UO}{F(U\Sigma)^{(*)} \to O} & \text{by Theorem 3.6} \\[2ex] \dfrac{F(U\Sigma)^{(*)} \to O}{(FU\Sigma)^* \to O} & \text{by Corollary 3.8} \\[2ex] \dfrac{(FU\Sigma)^* \to O}{\Sigma^* \to O} & \text{by Proposition 3.9} \end{array}$$

◀

Thus, every separated automaton over $U(\Sigma), U(O)$ gives rise to an Sb-language $\overline{S}$, corresponding to the language S accepted by the automaton.

Any nominal automaton $\mathcal{A}$ restricts to a separated automaton, formally described in Definition 4.7. It turns out that if the (Pm)-language accepted by $\mathcal{A}$ is actually an Sb-language, then the restricted automaton already represents this language, as the extension $\overline{S}$ of the associated separated language S (Proposition 4.8). Hence, in such a case, the restricted separated automaton suffices to describe the language of $\mathcal{A}$.

▶ **Definition 4.7.** *Let $i\colon Q * U(\Sigma) \hookrightarrow Q \times U(\Sigma)$ be the natural inclusion map. A nominal automaton $\mathcal{A} = (Q, \delta, o, q_0)$ induces a separated automaton $\mathcal{A}_*$, by setting $\mathcal{A}_* = (Q, \delta \circ i, o, q_0)$.*

▶ **Proposition 4.8.** *Suppose Σ, O are both* Sb*-sets, and suppose* $\dim(\Sigma) \leq 1$*. Let $L\colon (U\Sigma)^* \to UO$ be the* Pm*-nominal language accepted by a nominal automaton $\mathcal{A}$, and suppose L is* Sb*-equivariant. Let S be the separated language accepted by $\mathcal{A}_*$. Then $L = U(\overline{S})$.*

Proof. It follows from the one-to-one correspondence in Proposition 4.6: on the bottom there are two languages (L and $U(\overline{S})$), while there is only the restriction of L on the top. We conclude that $L = U(\overline{S})$. ◀

As we will see in Example 4.10, separated automata allow us to represent Sb-languages in a smaller way than nominal automata. Given a nominal automaton $\mathcal{A}$, a smaller separated automaton can be obtained by computing the reachable part of the restriction $\mathcal{A}_*$. The reachable part is defined similarly (but only where δ is defined) and also denoted by $R(\mathcal{A}_*)$.

▶ **Lemma 4.9.** *For any nominal automaton $\mathcal{A}$, we have $R(\mathcal{A}_*) \subseteq R(\mathcal{A})$.*

The converse inclusion of the above proposition does certainly not hold, as shown by the following example.

▶ **Example 4.10.** Let $\mathcal{A}$ be the automaton modelling a bounded FIFO queue (for some n), from Example 4.3. The Pm-nominal language L accepted by $\mathcal{A}$ is Sb-equivariant: it is closed under application of arbitrary substitutions.

The separated automaton $\mathcal{A}_*$ is given simply by restricting the transition function to $Q * \Sigma$, i.e., a $\mathsf{Put}(a)$-transition from a state $w \in Q$ exists only if a does not occur in w. The separated language S accepted by this new automaton is the restriction of the nominal language of $\mathcal{A}$ to separated words. By Proposition 4.8, we have $L = U(\overline{S})$. Hence, the separated automaton $\mathcal{A}_*$ represents L, essentially by closing the associated separated language S under all substitutions.

The *reachable* part of $\mathcal{A}_*$ is given by

$$R_{\mathcal{A}_*} = \mathbb{A}^{(\leq n)} \cup \{\bot\}\,.$$

Clearly, restricting $\mathcal{A}_*$ to the reachable part does not affect the accepted language. However, while the original state space Q has exponentially many orbits in n, $R_{\mathcal{A}_*}$ has only $n+2$ orbits! Thus, taking the reachable part of $R_{\mathcal{A}_*}$ yields a separated automaton which represents the FIFO language L in a much smaller way than the original automaton.

4.1 Separated automata: coalgebraic perspective

Nominal automata and separated automata can be presented as *coalgebras* on the category of Pm-nominal sets. In this section we revisit the above results from this perspective, and generalise from (equivariant) languages to finitely supported languages. In particular, we retrieve the extension from separated languages to Sb-languages, by establishing Sb-languages as a final separated automaton. The latter result follows by instantiating a well-known technique for lifting adjunctions to categories of coalgebras, using the results of Section 3. We assume familiarity with the theory of coalgebras, see, e.g., [14, 25].

▶ **Definition 4.11.** *Let M be a submonoid of Sb, and let Σ, O be nominal M-sets, referred to as the input and output alphabet respectively. Define the functor $B_M \colon M\text{-}\mathbf{Nom} \to M\text{-}\mathbf{Nom}$ by $B_M(X) = O \times (\Sigma \to^M_{\mathrm{fs}} X)$. An (M)*-nominal (Moore) automaton *is a B_M-coalgebra.*

A B_M-coalgebra can be presented as a nominal set Q together with the pairing

$$\langle o, \delta^\flat \rangle \colon Q \to O \times (\Sigma \to^M_{\mathrm{fs}} Q)$$

of an equivariant *output* function $o \colon Q \to O$, and (the transpose of) an equivariant *transition* function $\delta \colon Q \times \Sigma \to Q$. In case $M = \mathsf{Pm}$, this coincides with the automata of Definition 4.2, omitting initial states. The language semantics is generalised accordingly, as follows. Given such a B_M-coalgebra $(Q, \langle o, \delta^\flat \rangle)$, the *language semantics* $l \colon Q \times \Sigma^* \to O$ is given by

$$l(x, \varepsilon) = o(x)\,, \qquad l(x, aw) = l(\delta(x, a), w) \tag{4}$$

for all $x \in S$, $a \in \Sigma$ and $w \in \Sigma^*$.

▶ **Proposition 4.12.** *Let M be a submonoid of Sb, let Σ, O be nominal M-sets. The nominal M-set $\Sigma^* \to^M_{\mathrm{fs}} O$ extends to a final B_M-coalgebra $(\Sigma^* \to^M_{\mathrm{fs}} O, \zeta)$, such that the unique homomorphism from a given B_M-coalgebra is the transpose $l^\flat$ of the language semantics* (4).

A *separated automaton* (Definition 4.4, without initial states) corresponds to a coalgebra for the functor $B_* \colon \mathsf{Pm}\text{-}\mathbf{Nom} \to \mathsf{Pm}\text{-}\mathbf{Nom}$ given by $B_*(X) = O \times (\Sigma \nrightarrow X)$. The separated language semantics arises by finality.

▶ **Proposition 4.13.** *The set $\Sigma^{(*)} \multimap O$ is the carrier of a final B_*-coalgebra, such that the unique coalgebra homomorphism from a given B_*-coalgebra $(Q, \langle o, \delta \rangle)$ is the transpose $s^\flat$ of the separated language semantics $s \colon Q * \Sigma^{(*)} \to O$ (Definition 4.4).*

Next, we provide an alternative description of the final B_*-coalgebra which assigns Sb-nominal languages to states of separated nominal automata. The essence is to obtain a final B_*-coalgebra from the final B_{Sb}-coalgebra. In order to prove this, we use a technique to lift adjunctions to categories of coalgebras. This technique occurs regularly in the coalgebraic study of automata [15, 17, 16].

▶ **Theorem 4.14.** *Let Σ be a Pm-set, and O an Sb-set. Define B_* and B_{Sb} accordingly, as $B_*(X) = UO \times (\Sigma \multimap X)$ and $B_{\mathsf{Sb}}(X) = O \times (F\Sigma \to^{\mathsf{Sb}}_{\mathrm{fs}} X)$. There is an adjunction*

$$\mathrm{CoAlg}(B_*) \underset{\overline{U}}{\overset{\overline{F}}{\rightleftarrows}} \mathrm{CoAlg}(B_{\mathsf{Sb}}) \quad (\perp)$$

where $\overline{F}$ and $\overline{U}$ coincide with F and U respectively on carriers.

Proof. There is a natural isomorphism $\lambda \colon B_*U \Rightarrow UB_{\mathsf{Sb}}$ given by

$$\lambda \colon UO \times (\Sigma \multimap UX) \xrightarrow{\mathrm{id} \times \phi} UO \times U(F\Sigma \to^{\mathsf{Sb}}_{\mathrm{fs}} X) \xrightarrow{\cong} U(O \times (F\Sigma \to^{\mathsf{Sb}}_{\mathrm{fs}} X)) \,,$$

where ϕ is the isomorphism from Theorem 3.13 and the isomorphism on the right comes from U being a right adjoint. The result now follows from Theorem 2.14 in [12]. In particular, $\overline{U}(X, \gamma) = (UX, \lambda^{-1} \circ U(\gamma))$. ◀

Since right adjoints preserve limits, and final objects in particular, we obtain the following, giving semantics of separated automata through finality.

▶ **Corollary 4.15.** *Let $((F\Sigma)^* \to^{\mathsf{Sb}}_{\mathrm{fs}} O, \zeta)$ be the final B_{Sb}-coalgebra (Proposition 4.12). Then the B_*-coalgebra $\overline{U}(\Sigma^* \to^{\mathsf{Sb}}_{\mathrm{fs}} O, \zeta)$ is final and carried by the set $(F\Sigma)^* \to^{\mathsf{Sb}}_{\mathrm{fs}} O$ of Sb-nominal languages.*

5 Relation to (pre)sheaf categories

Fiore and Turi described a similar adjunction between certain presheaf categories [7]. However, Staton describes in his thesis that the usage of presheaves allows for many degenerate models and one should look at sheaves instead [29]. The category of sheaves is equivalent to the category of nominal sets. We will describe these equivalences in this section.

Let us define the index categories $\mathbb{I}$ and $\mathbb{F}$. Both categories have finite subsets $C \subset \mathbb{A}$ as objects. The morphisms in $\mathbb{I}$ are all injective functions between those sets, and for $\mathbb{F}$ we take all functions. The presheaves Fiore and Turi considered are $\mathbf{Set}^{\mathbb{I}}$ and $\mathbf{Set}^{\mathbb{F}}$. The interpretation of an object $X \in \mathbf{Set}^{\mathbb{I}}$ is that $X(C)$ is the set of elements supported by C. Although very similar to nominal sets, the categories are not equivalent. The inclusion $\mathbb{I} \subseteq \mathbb{F}$ induces a forgetful functor $\mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{I}}$. It has a left adjoint, which can be defined by a Kan extension [7].

The subcategory of functors $\mathbb{I} \to \mathbf{Set}$ which preserve pullbacks is a sheaf category $\mathbf{Sh}(\mathbb{I})$.[5] (For the precise sheaf conditions, see Staton's thesis [29].) The category $\mathbf{Sh}(\mathbb{I})$ is equivalent to $\mathbf{Nom}$. Similarly, there is a sheaf category $\mathbf{Sh}(\mathbb{F}) \subseteq \mathbf{Set}^{\mathbb{F}}$ and Staton has shown that the adjunction $\mathbf{Set}^{\mathbb{I}} \rightleftarrows \mathbf{Set}^{\mathbb{F}}$ restricts to an adjunction $\mathbf{Sh}(\mathbb{I}) \rightleftarrows \mathbf{Sh}(\mathbb{F})$.

[5] We use the notation from [29], since we only deal with covariant functors here.

How does this compare to the adjunction described in this paper? Staton defines a category of nominal sets with a substitution operator, defined by certain axioms. This fits in the theory of universal algebra on nominal sets, as described by Kurz and Petrişan [18]. This category **NomSub** is equivalent to $\mathbf{Sh}(\mathbb{F})$, and most likely equivalent to Sb-**Nom** as defined here. These equivalences then give an abstract way of defining the adjunction from Theorem 3.6. Together with the fact that the separated product is a Day convolution (this fact is hinted at in [5] and [21]), one might obtain Theorem 3.7 from abstract reasoning alone (using the fact that both the left adjoint and the separated product are left Kan extensions). Nevertheless, we think that the explicit constructions and proofs given in this paper are useful, as they provide a concrete interpretation of the abstract concepts.

6 Related and future work

An interesting line of research is the generalisation to other symmetries by Bojańczyk et al. [3]. In particular, the total order symmetry is relevant, since it allows one to compare elements on their order, as often used in data words. In this case the symmetries are given by the group of all monotone bijections. Many results of nominal sets generalise to this symmetry. For monotone substitutions, however, the situation seems more subtle. For example, we note that a substitution which maps two values to the same value actually maps *all* the values in between to that value. Whether the adjunction from Theorem 3.6 generalises to other symmetries is left as future work.

This research was motivated by learning register automata. If we know a register automaton recognises an Sb-language, then we are better off learning a separated automaton instead of a nominal automaton. From the Sb-semantics of separated automata, it follows that we have a Myhill-Nerode theorem, which means that learning is feasible. We expect that this can be useful, since we can achieve an exponential reduction this way.

Bojańczyk et al. prove that nominal automata are equivalent to register automata in terms of expressiveness [3]. However, when translating from register automata with n states to nominal automata, we may get exponentially many orbits. This happens for instance in the FIFO automaton (Example 4.3). We have shown that the exponential blow-up is avoidable by using separated automata, for this example and in general for Sb-equivariant languages. Such languages come from register automata which manipulate data but where do control flow does not depend on comparisons. This typically occurs in data structures.

An important open problem is whether the latter requirement can be relaxed, by adding separated transitions only locally in a nominal automaton. A possible step in this direction is to consider the monad $T = UF$ on Pm-**Nom** and incorporate it in the automaton model. We believe that this is the hypothesised "substitution monad" from [20]. The monad is monoidal (sending separated products to Cartesian products) and if X is an orbit-finite nominal set, then so is $T(X)$. This means that we can consider nominal T-automata and we can perhaps determinise them using coalgebraic methods [27].

References

1 Michael Francis Atiyah and Ian G. MacDonald. *Introduction to commutative algebra*. Addison-Wesley-Longman, 1969.

2 Mikołaj Bojańczyk. *Slightly Infinite Sets*. Draft May 22, 2019. URL: `https://www.mimuw.edu.pl/~bojan/upload/main-9.pdf`.

3 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014. `doi:10.2168/LMCS-10(3:4)2014`.

4 Mikołaj Bojańczyk, Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk. Turing Machines with Atoms. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 183–192. IEEE Computer Society, 2013. doi:10.1109/LICS.2013.24.

5 Ranald Clouston. Generalised Name Abstraction for Nominal Sets. In Frank Pfenning, editor, *Foundations of Software Science and Computation Structures - 16th International Conference, FOSSACS 2013. Proceedings*, volume 7794 of *Lecture Notes in Computer Science*, pages 434–449. Springer, 2013. doi:10.1007/978-3-642-37075-5_28.

6 Gilles Dowek and Murdoch James Gabbay. PNL to HOL: from the logic of nominal sets to the logic of higher-order functions. *Theor. Comput. Sci.*, 451:38–69, 2012. doi:10.1016/j.tcs.2012.06.007.

7 Marcelo P. Fiore and Daniele Turi. Semantics of Name and Value Passing. In *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*, pages 93–104. IEEE Computer Society, 2001. doi:10.1109/LICS.2001.932486.

8 Murdoch J. Gabbay. FM-HOL, a higher-order theory of names. *Thirty Five years of Automath, Heriot-Watt University, Edinburgh*, 2002.

9 Murdoch J. Gabbay. Nominal Renaming Sets. Technical report, Heriot-Watt University, 2007. URL: https://www.gabbay.org/paper.html#nomrs-tr.

10 Murdoch J. Gabbay and Martin Hofmann. Nominal Renaming Sets. In Iliano Cervesato, Helmut Veith, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 15th International Conference, LPAR 2008. Proceedings*, volume 5330 of *Lecture Notes in Computer Science*, pages 158–173. Springer, 2008. doi:10.1007/978-3-540-89439-1_11.

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

12 Claudio Hermida and Bart Jacobs. Structural Induction and Coinduction in a Fibrational Setting. *Inf. Comput.*, 145(2):107–152, 1998. doi:10.1006/inco.1998.2725.

13 Malte Isberner, Falk Howar, and Bernhard Steffen. Learning register automata: from languages to program structures. *Machine Learning*, 96(1-2):65–98, 2014. doi:10.1007/s10994-013-5419-7.

14 Bart Jacobs. *Introduction to Coalgebra: Towards Mathematics of States and Observation*, volume 59 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2016. doi:10.1017/CBO9781316823187.

15 Bart Jacobs, Alexandra Silva, and Ana Sokolova. Trace semantics via determinization. *J. Comput. Syst. Sci.*, 81(5):859–879, 2015. doi:10.1016/j.jcss.2014.12.005.

16 Henning Kerstan, Barbara König, and Bram Westerbaan. Lifting Adjunctions to Coalgebras to (Re)Discover Automata Constructions. In Marcello M. Bonsangue, editor, *Coalgebraic Methods in Computer Science - 12th IFIP WG 1.3 International Workshop, CMCS 2014, Colocated with ETAPS 2014, Grenoble, France, April 5-6, 2014, Revised Selected Papers*, volume 8446 of *Lecture Notes in Computer Science*, pages 168–188. Springer, 2014. doi:10.1007/978-3-662-44124-4_10.

17 Bartek Klin and Jurriaan Rot. Coalgebraic trace semantics via forgetful logics. *Logical Methods in Computer Science*, 12(4), 2016. doi:10.2168/LMCS-12(4:10)2016.

18 Alexander Kurz and Daniela Petrisan. On universal algebra over nominal sets. *Mathematical Structures in Computer Science*, 20(2):285–318, 2010. doi:10.1017/S0960129509990399.

19 Joshua Moerman and Jurriaan Rot. Separation and Renaming in Nominal Sets. *CoRR*, abs/1906.00763, 2019.

20 Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. Learning nominal automata. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL*

2017, Paris, France, January 18-20, 2017, pages 613–625. ACM, 2017. URL: http://dl.acm.org/citation.cfm?id=3009879.

21 Peter W. O'Hearn. On bunched typing. *J. Funct. Program.*, 13(4):747–796, 2003. doi:10.1017/S0956796802004495.

22 Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013.

23 Andrew M. Pitts. Nominal Presentation of Cubical Sets Models of Type Theory. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs and Programs, TYPES 2014, May 12-15, 2014, Paris, France*, volume 39 of *LIPIcs*, pages 202–220. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.TYPES.2014.202.

24 Andrew M. Pitts. Nominal techniques. *SIGLOG News*, 3(1):57–72, 2016. doi:10.1145/2893582.2893594.

25 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

26 Ulrich Schöpp. *Names and binding in type theory*. PhD thesis, University of Edinburgh, UK, 2006. URL: http://hdl.handle.net/1842/1203.

27 Alexandra Silva, Filippo Bonchi, Marcello M. Bonsangue, and Jan J. M. M. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013. doi:10.2168/LMCS-9(1:9)2013.

28 Harold Simmons. The topos of actions on a monoid. Unpublished manuscript, number 12N. URL: http://www.cs.man.ac.uk/~hsimmons/DOCUMENTS/PAPERSandNOTES/Rsets.pdf.

29 Sam Staton. *Name-passing process calculi: operational models and structural operational semantics*. PhD thesis, University of Cambridge, Computer Laboratory, June 2007. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-688.pdf.