# A Name Abstraction Functor for Named Sets

Vincenzo Ciancia [1] Ugo Montanari [1]

*Department of Computer Science*
*University of Pisa*

Abstract

The problem of defining fully abstract operational models of name passing calculi has been given some elegant solutions, such as coalgebras over presheaf categories or over nominal sets. These formalisms fail to model garbage collection of unused names, hence they do not have nice properties with respects to finite state algorithms. The category of named sets, on the other hand, was designed for the purpose of supporting efficient algorithms to handle the semantics of name passing calculi. However the theory was developed in a rather *ad-hoc* fashion (e.g. the existence of a final coalgebra was only proved in the finite case). In this work we introduce a name abstraction functor for named sets and show that it provides a simple and effective notion of garbage collection of unused names. Along the way, we survey a number of needed results on the category of permutation algebras, an algebra-theoretic definition of nominal sets. In particular we give a formalization of the adjunction between abstraction and concretion, an example illustrating a *nominal syntax alike* handling of De Bruijn indexes, and an explicit functor to model the early semantics of the $\pi$-calculus in nominal sets.

*Keywords:* named sets, nominal sets, binding, garbage collection, De Bruijn indexes

## 1 Introduction

In the field of programming language semantics and concurrency theory, wide attention is paid nowadays to the so called *name-passing calculi*, i.e. formalisms where name generation and passing play a fundamental role. Among the most important research contributions in this area, we find the introduction of the $\pi$-calculus [1], which allows one to model the topology of a network of mobile components, by generating and passing *fresh* channels. Traditional logical methods, such as modal logics, proved inadequate to reason about this formalism, leading to the definition of logics with name binding, such as [2] or [3].

On the side of the syntax, the problem of representing binding in a satisfying way has been long standing, the most famous attempt being that of *De Bruijn indexes* [4]. The problem was tackled in the work on *nominal syntax* [5,6,7], aimed to provide a logical framework in which to represent axioms on terms up to $\alpha$-equivalence [8], employing the category of *nominal sets*. Also on the side of semantic models, it was soon recognized that ordinary labeled transition systems (LTS) and their

[1] Research partially supported by the EU FP6-IST IP 16004 SENSORIA

*This paper is electronically published in*
*Electronic Notes in Theoretical Computer Science*
*URL:* www.elsevier.nl/locate/entcs

bisimulations do not fully capture the notion of name allocation, which is required in the operational semantics of the $\pi$-calculus (and other name-passing calculi). New models where developed, noticeably *presheaf* models, such as [9,10], and those based on the theory of *permutation algebras* [11,12].

As it was soon recognized, even though they where developed independently, the category of nominal sets, and that of permutation algebras, are essentially the same. We remark that the latter is not a new definition, but rather the standard notion of algebra for a specification. While the idea of modeling binding in this category certainly has its roots in the work on nominal syntax, the key role of the symmetry of elements, which subsumes the notion of *finite support* and its properties (cf. theorem 2.11 in the following) is a result of the research on permutation algebras. A key motivation for studying symmetries was the development of *history-dependent automata with symmetries* [13], a categorial model, based on the theory of *named sets*, closely connected to permutation algebras, capable of finite-state verification of processes that can allocate fresh resources.

A comparison of the various categorial models of name passing was done in [14] and [15], resulting in the proof of categorial equivalence of permutation algebras (hence nominal sets), the Schanuel topos (the pullback-preserving full subcategory of the presheaf category $\mathbf{Set}^{\mathbf{I}}$) and named sets. Even though this shows that, from a theoretical point of view, there is no difference in adopting one of a number of equivalent formalisms, named sets enjoy nice algorithmic properties which are absent in all the other models, and are due to the handling of names as *local*, bindable resources, rather than constants which have a *global*, immutable meaning.

The equivalence result in [14] was focused on the base categories, without analyzing in detail functors and associated categories of *coalgebras* that express the operational semantics of calculi. In this work, along the way to define a name abstraction functor on named sets, we add some missing pieces to this puzzle.

In [12], the final coalgebra semantics of the $\pi$-calculus was obtained in an implicit form, reusing results on lifting of SOS rules [16] defining transition systems in **Set** (again, exploiting the algebraic definition of the category of nominal sets). The functor defined by the lifting was never made explicit, thus name abstraction was never defined for permutation algebras. We do this exploiting a theory morphism.

A coalgebra modeling the early semantics of the $\pi$-calculus using named sets was given in [17,18], together with a minimization procedure based on *iteration along the terminal sequence*. However, the functor given there was defined in an *ad-hoc* way, without providing separate components (such as name abstraction) that allow one to reuse the existing theory to express the semantics of other calculi. The "monolithic" definition of the functor therein makes difficult to reason about it (for example, it has not been proved if it has admits a final coalgebra).

Our main contribution is to develop the equivalence results in [14] a step further, by characterizing, in the category of named sets, the *name abstraction* functor modeling binding operations. Name abstraction can be considered, not without a reason, the novel, and most important feature of the name-aware models we mentioned, and we foresee that it will allow us a generalization of the results in [18].

The map of the paper is as follows. We start by defining permutation algebras in section 2.1, and proving a number of relevant (possibly folklore) results in section

2.2. Among these, without claiming originality, we mention some that up to our knowledge had not been clearly stated before, even though deserving attention: theorem 2.13, and theorem 2.11, whose subsequent corollary is one of the peculiar features of permutation algebras (and nominal sets).

Then, in section 3, exploiting the notion of theory morphism (whose definition is recalled in section 2.3), we obtain the abstraction functor $\delta$ on permutation algebras as the *forgetful* functor of the associated adjunction. The corresponding *free* functor $F$ is called *concretion*. Group-theoretical properties (namely, orbits and symmetries) of both functors are explored in sections 4.1 and 4.2.

In section 4.3 we provide the due comparison with the *nominal set of abstractions* and the *freshness relation* of nominal sets. The notions turn out to be isomorphic (hence, we get a proof of another folklore result, the adjunction between the two functors in nominal syntax). Nevertheless, the adjunction we provided is still worth being studied for some side results. In example 4.12, we provide the definition of the machinery of De Bruijn indexes as the initial algebra for a functor involving $\delta$, thus filling the gap between nominal abstract syntax and De Bruijn indexes. Moreover, the pair of adjoints introduces the counit, whose effect is to *reveal* a bound name, in a similar way to the *concretion operation* of nominal syntax. Giving a precise role to this operator opens the way to formally link the field of spatial logics for nominal calculi [19,20] with the algorithmic framework of named sets (see remark 4.13).

On the side of the semantics of calculi, we show in section 5 that the functor used in [12] can be defined using ordinary endofunctors (polynomials and power set) and $\delta$, thus addressing the question of an explicit form for that functor. The definition of coalgebraic models over nominal sets has been recently proposed also in [21], where an example showing the *late* semantics of the $\pi$-calculus is provided using the abstraction functor of nominal syntax.

In section 6 we introduce the theory of named sets, attempting to explain some of their properties in terms of results on permutation algebras, in light of the equivalence result of [14] which we shortly present. Finally, in section 7.1, we define the abstraction functor in named sets, exploiting results of section 4.1. An isomorphism result with the corresponding functor in permutation algebras (thus, with abstraction as defined in nominal syntax) is established.

The obtained framework has a distinguishing feature, with respects to presheaves and permutation algebras: the abstraction functor has an implicit notion of *garbage collection*, that allows names to be discarded, without necessarily keeping memory of names generated in the past transitions. This is not the case, either in presheaves or in permutation algebras/nominal sets, as we explain in section 7.2.

## 2 Background

Here we introduce the basic algebraic and group-theoretic notions that are used in the rest of the paper. In section 2.1 we introduce the category of sets equipped with an *action* of finite-kernel permutations over the set of natural numbers $\omega$, and corresponding morphisms. These are usually known as FM-sets from the set theory of Fraenkel and Mostowski, who developed this model to show independence of the *axiom of choice* from other axioms of set theory. This category can be conveniently

defined observing that FM-sets are algebras for the finite-kernel permutation group viewed as a monadic, single-sorted algebraic theory, obtaining the notion of *equivariant function* as the morphisms of the associated category of algebras. In section 2.2, we show a number of results that are needed. Many of these are folklore, the remarkable exception being theorem 2.11, a simple result that explains in terms of symmetry one of the most pervasive properties of the support: it never grows along morphisms. This leads, in all categorial formalisms that handle names using injective relabelings (presheaves, nominal sets and named sets), to the necessity of defining specialized functors for abstraction. In section 2.3 the notion of theory morphism and associated adjunction is explained.

### *2.1 Nominal Sets as Algebras for the Permutation Signature*

The set of permutations over a set $S$ forms a group where the operation is function composition. The core of the theory we present is the subgroup of *finite-kernel* permutations over the set of natural numbers $\omega$, which we will denote with $Autf$, without making an explicit distinction between it seen as a set, a group, or the algebraic specification of definition 2.2 below.

**Definition 2.1** *(finite-kernel permutation) The* kernel $ker(\pi)$ *of a permutation* $\pi : N \to N$ *is the set* $\left\{x \in N \middle| \pi(x) \neq x\right\}$. *A permutation is* finite-kernel *if its kernel is finite. The set of all finite-kernel permutations over* $\omega$ *is denoted with* $Autf$.

Now we introduce the notions of permutation signature [**] and permutation algebra.

**Definition 2.2** *(permutation algebra) The (finite-kernel)* permutation signature *over* $\omega$ *is the set* $Autf$ *of finite kernel permutations over* $\omega$, *considered as* unary, one-sorted *algebraic operations of the form* $\pi : 1 \to 1$, *together with the equational axioms* $id(x) = x$ *and* $\pi_1(\pi_2(x)) = (\pi_1 \circ \pi_2)(x)$ *for each* $\pi_1$ *and* $\pi_2$ *in* $Autf$. *A* permutation algebra *is an algebra* $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}} : A \to A \mid \pi \in Autf\}\rangle$ *for* $Autf$, *where* $A$ *is the carrier set, and* $\pi_{\mathcal{A}}$ *is the interpretation of* $\pi$, *also called* permutation action.

Identity and composition operations are not explicitly part of the permutation signature. However the axiom for identity, and the axiom schema for composition, make the interpretation of composed permutations act like composition.

**Definition 2.3** *(morphism) An* algebra morphism *between permutation algebras* $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$ *and* $\mathcal{B} = \langle B, \{\pi_{\mathcal{B}}\}\rangle$ *is a function* $f : A \to B$ *that preserves operations of the permutation signature, i.e.* $\forall \pi \in Autf\ .f(\pi_{\mathcal{A}}(x)) = \pi_{\mathcal{B}}(f(x))$.

Such morphisms are called *equivariant functions* in the theory of nominal syntax. Notice that $\omega$ can be considered a permutation algebra using the natural interpretation $\pi_\omega = \pi$. We now introduce the group-theoretical notions of *symmetry* (also called *isotropy group* or *stabilizer*), *support* and *orbit*.

**Definition 2.4** *(symmetry) The* symmetry *of an element* $a$ *of* $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$ *is the set of all permutations fixing* $a$ *in* $\mathcal{A}$, *defined as* $\mathcal{G}_{\mathcal{A}}(a) = \{\pi \in Autf \mid \pi_{\mathcal{A}}(a) = a\}$.

[**] The definition really introduces an algebraic specification made up of a signature and an equational theory, not just a signature. However, since "permutation signature" has often been used to denote that theory, we stick with common terminology.

The identity group of $X \subseteq \omega$ is defined as $fix(X) = \{\pi \in Autf \mid \pi_{|_X} = id_X\}$ and is used to define the notion of *support* in permutation algebras.

**Definition 2.5** *(support) We say that* $X \subseteq \omega$ supports $a \in A$ *if* $fix(X) \subseteq \mathcal{G}_{\mathcal{A}}(a)$*, i.e. if all permutations fixing* $X$ *also fix* $a$ *in* $\mathcal{A}$*. The least* finite *set* $X$ *satisfying this condition, if it exists, is called the* support *of* $a$*, written* $supp_{\mathcal{A}}(a)$*. A permutation algebra is said to be* finitely supported *if all of its elements have finite support.*

Each element of a permutation algebra is trivially supported by $\omega$. A *finite* supporting set might not exist, but if there is one, then the support is the intersection of all of them. The notion of support generalizes that of "free variables" of terms, thus we will often refer to $\omega$ as the set of *names*. On the other hand, the notion of symmetry models *indistinguishability* of free names with respects to certain permutations.

**Definition 2.6** *(category of permutation algebras) Permutation algebras and their morphisms form a category, named* $\mathbf{Alg}^{\pi}$*. We will denote with* $\mathbf{FSAlg}^{\pi}$ *the full subcategory of finitely supported permutation algebras and their morphisms.*

Finitely-supported permutation algebras are the *nominal sets* of Gabbay and Pitts. The last notion we need to introduce on permutation algebras is that of *orbit*.

**Definition 2.7** *(orbit) The* orbit *of* $a \in A$ *is* $orb_{\mathcal{A}}(a) = \{\pi_{\mathcal{A}}(a) \mid \pi \in Autf\}$.

Orbits partition algebras in equivalence classes. We denote with $a^o_{\mathcal{A}}$ the (canonical representative of) the equivalence class of $a$, and with $X^o_{\mathcal{A}}$ the set $\{x^o_{\mathcal{A}} \mid x \in X\}$, for $X \subseteq A$. Orbits play a central role when switching from the category of permutation algebras to their "finitistic" counterpart, *named sets*.

**Example 2.8** *(terms with variables) Let* $\Sigma$ *be a signature. Free terms with variables in* $\omega$ *form a permutation algebra* $\mathcal{T} = \langle T_{\Sigma}(\omega), \{\pi_{\mathcal{T}}\}\rangle$*, with* $\pi_{\mathcal{T}}(t) = t\left[{}^{\pi(i)}/_{i}\right]$ *for* $i \in \omega$*. It is easy to see that a finite set* $X \subseteq \omega$ *supports a term* $t$ *if and only if its set of free variables* $FV(t)$ *is a subset of* $X$*. So, the least such set, i.e. the support, is the set of free variables of* $t$*.*

### *2.2 Some Results on Permutation Algebras*

First of all, we show that the action of a permutation $\pi$ on elements of a permutation algebra is determined by the action of $\pi$ on their support.

**Lemma 2.9** *In a finitely supported permutation algebra* $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$*, for each* $a \in A$*, we have* $\pi_{|_{supp_{\mathcal{A}}(a)}} = \pi'_{|_{supp_{\mathcal{A}}(a)}} \implies \pi_{\mathcal{A}}(a) = \pi'_{\mathcal{A}}(a)$.

**Remark 2.10** *Because of lemma 2.9, we will usually define a permutation* $\pi$ *only on the support of an element* $a \in A$*, when it is clear from the context that* $\pi$ *is to be applied only to* $a$*. In this case, we assume that the definition of* $\pi$ *is* completed *in order to obtain a finite-kernel permutation*

The following theorem, and in particular its subsequent corollary, gives an important property of nominal sets, extensively used in proofs about this category. Even though the corollary is well known (see e.g. corollary 9 in [14]), it is interesting to observe that it just comes from the symmetry of elements.

**Theorem 2.11** *Let* $f : \langle A, \{\pi_{\mathcal{A}}\}\rangle \to \langle B, \{\pi_{\mathcal{B}}\}\rangle$*. Then* $\forall a \in A. \mathcal{G}_A(a) \subseteq \mathcal{G}_B(f(a))$.

**Lemma 2.12** *For each morphism $f$ in* $\mathbf{FSAlg}^{\pi}$ *, we have* $supp_{\mathcal{B}}(f(a)) \subseteq supp_{\mathcal{A}}(a)$.

The following "isomorphism theorem" is relevant for named sets, since it asserts that a named set represents a class of isomorphic permutation algebras.

**Theorem 2.13** *Two permutation algebras* $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$ *and* $\mathcal{B} = \langle B, \{\pi_{\mathcal{B}}\}\rangle$ *are isomorphic if and only if there exists a choice of canonical representatives of orbits, and an isomorphism* $i : A^o_{\mathcal{A}} \to B^o_{\mathcal{B}}$ *in* **Set**, *such that* $\mathcal{G}_{\mathcal{A}}(a^o) = \mathcal{G}_{\mathcal{B}}(i(a^o))$.

Finally, we provide a "representation theorem", taken from [12], aimed to give a finite representation of the symmetry of finitely supported permutation algebras.

**Theorem 2.14** *The symmetry* $\mathcal{G}_{\mathcal{A}}(a)$ *of* $a \in A$ *is obtained by composition of two subgroups as follows:* $\mathcal{G}_{\mathcal{A}}(a) = fix(supp_{\mathcal{A}}(a)) \circ (\mathcal{G}_{\mathcal{A}}(a) \cap fix(\omega \setminus supp_{\mathcal{A}}(a)))$

In words, the infinite set of all permutations in $\mathcal{G}_{\mathcal{A}}(a)$ can be reconstructed from the information described by the (finite) set of all permutations in $\mathcal{G}_{\mathcal{A}}(a)$ that only alter the support of $a$, by composition with *all* the permutations that only alter names *outside* the support of $a$. This theorem is exploited in named sets to obtain a finite description of the symmetries. Such a finite description is still a group, hence it can be *efficiently* represented (using its generators, see [18]).

### *2.3 Theory Morphisms*

Theory morphisms, or *views*, are equation-preserving signature morphisms $M : \Sigma_1 \to \Sigma_2$, that yield algebras of $\Sigma_1$ from algebras of $\Sigma_2$. Here we just deal with the single-sorted case, since it is sufficient to present our work. The contents of this section are standard material from the theory of algebras (see e.g. [22]).

We denote with $T_\Sigma$ the initial algebra of the signature $\Sigma$, with $T_\Sigma(V)$ the free $\Sigma$-algebra over a set of variables $V$, and with $T_{\Sigma,E}(V)$ the free $\Sigma$-algebra over $V$ quotiented with equations *derivable* from $E$. The operations of the two initial algebras are indicated with $op_{T_\Sigma(V)}$ and $op_{T_{\Sigma,E}(V)}$, for $op \in \Sigma$. Given a $\Sigma$-algebra $\mathcal{A} = \left\langle A, \left\{ op_{\mathcal{A}} \middle| op \in \Sigma \right\} \right\rangle$, we call *presentation* of $\mathcal{A}$, denoted with $Pres(\mathcal{A})$, the *kernel* of the unique $m : T_\Sigma \to \mathcal{A}$, i.e. the set of pairs $t_1 = t_2$ such that $t_1, t_2 \in T_\Sigma$ and $m(t_1) = m(t_2)$. $Eq(E)$ represents the set of all equations derivable from $E$.

**Definition 2.15** *A* signature morphism $M$ *between signatures* $\Sigma_1$ *and* $\Sigma_2$ *is a function from the operators of* $\Sigma_1$ *to the operators of* $\Sigma_2$ *that respects operator arity, i.e. for every operator op of arity k,* $M(op)$ *has arity k. A signature morphism is inductively extended to* $T_\Sigma(V)$, *as* $M(op(T_1, \ldots, T_k)) = M(op)(M(T_1), \ldots, M(T_k))$, *and* $M(X \in V) = X$, *and to equations as* $M(T_1 = T_2) = (M(T_1) = M(T_2))$. *Given two specifications* $S_1 = \langle \Sigma_1, E_1 \rangle$ *and* $S_2 = \langle \Sigma_2, E_2 \rangle$, *a* theory morphism *from* $S_1$ *to* $S_2$ *is a signature morphism from* $\Sigma_1$ *to* $\Sigma_2$ *that preserves equations derivable from* $E_1$, *i.e.* $(T_1 = T_2 \in Eq(E_1)) \implies (M(T_1 = T_2) \in Eq(E_2))$.

Every theory morphism induces a (forgetful) functor from the category of algebras of its destination to the category of algebras of its source, having a left adjoint.

**Definition 2.16** *Let* $Th_1 = \langle \Sigma_1, E_1 \rangle$ *and* $Th_2 = \langle \Sigma_2, E_2 \rangle$ *be two specifications. A* theory morphism $M : Th_1 \to Th_2$ *associates to every* $\langle \Sigma_2, E_2 \rangle$*-algebra* $\mathcal{A} = \langle A, \{op_{\mathcal{A}} \mid op \in \Sigma_2\} \rangle$ *a* $\langle \Sigma_1, E_1 \rangle$*-algebra* $U(\mathcal{A}) = \langle A, \{op_{U(\mathcal{A})} \mid op \in \Sigma_1\} \rangle$ *with the*

*same carrier, where* $op_{U(\mathcal{A})} = M(op)_{\mathcal{A}}$ *for each operator* $op$ *in* $\Sigma_1$. *The map* $U$ *extends to a functor, acting on arrows as* $U_{arr}(f) = f$. $U$ *has a left adjoint.*

A definition of the left adjoint $F$ of $U$ can be given as a free construction that returns, for each $\Sigma_1$-algebra $\mathcal{A}$, the free $\Sigma_2$-algebra over its carrier $A$, quotiented with the translation of $Pres(\mathcal{A})$ via $M$.

**Definition 2.17** *Given a theory morphism* $M : \langle \Sigma_1, E_1 \rangle \to \langle \Sigma_2, E_2 \rangle$, *for each* $\langle \Sigma_1, E_1 \rangle$*-algebra* $\mathcal{A} = \langle A, \{op_{\mathcal{A}}\} \rangle$, *there is an associated* $\langle \Sigma_2, E_2 \rangle$*-algebra* $F(\mathcal{A}) = \langle T_{\Sigma_2, E_2}(A) /_{\equiv}, \{op_{T_{\Sigma_2, E_2}(A)}\} \rangle$, *where* $\equiv$ *is the equivalence relation defined by*

$$\frac{T_1 = T_2 \in Pres(\mathcal{A})}{M(T_1) \equiv M(T_2)}$$

*This map extends to a functor, acting on arrows* $f : \langle A, \{op_{\mathcal{A}}\} \rangle \to \langle B, \{op_{\mathcal{B}}\} \rangle$ *as* $F(f)(x \in A) = f(x)$, $F(f)(op(T_1, \ldots, T_n)) = op(F(f)(T_1), \ldots, F(f)(T_n))$.

**Theorem 2.18** $F$ *is left adjoint to* $U$.

# 3 Name Abstraction and Concretion

In this section, we introduce the forgetful functor $\delta$ induced by a particular theory morphism, and its left adjoint $F$. We call these *abstraction* and *concretion* in analogy with the terminology of nominal syntax. We show how the unit and the counit of the adjunction are obtained, deferring informal explanations to section 4.1. The definitions here and those of [7] may look different, but obtained objects are easily shown to be isomorphic. Our contribution here is to show how, by viewing the model as a category of algebras, we can characterize these two functors and their adjunction as a standard result.

## *3.1 The Abstraction and Concretion Functors*

A signature morphism from *Autf* to itself is a function that preserves identity and composition. We now define such a morphism.

**Definition 3.1** *The* right shift *operator* $-^{+1}$ :*Autf*$\to$*Autf gives, for each operation* $\pi$, *the operation* $\pi^{+1}$ *such that* $\pi^{+1}(0) = 0$, $\pi^{+1}(i+1) = \pi(i) + 1$.

**Theorem 3.2** *The right shift operator is a theory morphism.*

The purpose of the morphism is to send a permutation $\pi$ into $\iota \circ (\pi \oplus 1) \circ \iota^{-1}$, where $\iota$ is an isomorphism from $\omega \oplus 1$ to $\omega$. We have chosen the isomorphism $\hat{\iota}$ sending $\langle 0, * \rangle$ to 0 and $\langle 1, i \rangle$ to $i + 1$ (the famous "Hilbert's hotel paradox"), but any other choice is possible[⋆⋆⋆]. We can describe all these different $\iota$ as the set $\{\rho \circ \hat{\iota} \mid \rho \in Autf\}$. The effect of a generic $\iota$ is to "make room" for a fresh name, that actually comes from $\omega \oplus 1$. This makes evident the close analogy with the "typed" counterpart of nominal sets: presheaf categories such as $\mathbf{Set}^{\mathbf{I}}$, where the abstraction functor is usually defined on a presheaf $T$ as $\delta(T)(n) = T(n \oplus 1)$.

We obtain the *abstraction* functor as the associated forgetful functor.

⋆⋆⋆The choice we make matches the idea of De Bruijn indexes, as shown in example 4.12.

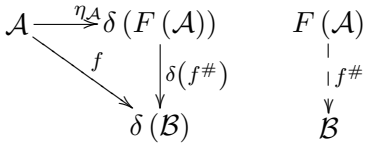


Figure 1.

**Definition 3.3** *The* name abstraction *functor* $\delta : \mathbf{Alg}^{\pi} \to \mathbf{Alg}^{\pi}$ *acts on objects as* $\delta_{obj}(\langle A, \{\pi_{\mathcal{A}}\}\rangle) = \langle A, \{\pi_{\mathcal{A}}^{+1}\}\rangle$, and on arrows as $\delta_{arr}(f) = f$.

Being defined by a theory morphism, $\delta$ has a left adjoint, the *concretion* functor. The permutation signature is only made up of unary, composable operators, so the presentation of a permutation algebra is in the simple form $\pi(a) = \rho(b)$, for $a, b \in A$, and $\pi, \rho \in Autf$. In the following definition, we represent the free algebra over $A$ simply as the product $A \times Autf$, i.e. we write $\langle a, \pi\rangle$ instead of the usual notation $\pi(a)$ for terms in $T_{Autf}(A)$, to avoid confusion with the notation $\pi_{\mathcal{A}}(a)$. Notice that terms in $T_{Autf}(A)$ are already quotiented with axioms of the permutation signature.

**Definition 3.4** *The functor* $F : \mathbf{Alg}^{\pi} \to \mathbf{Alg}^{\pi}$ *which is left adjoint to* $\delta$ *is defined on objects as* $F_{obj}(\langle A, \{\pi_A\}\rangle) = \left\langle T_{Autf}(A)/_{\equiv}, \pi_{F(\mathcal{A})}\right\rangle$, *with* $\pi_{F(\mathcal{A})}(\langle a, \rho\rangle) = \langle a, \pi \circ \rho\rangle$, *and on arrows as* $F_{arr}(f)(\langle a, \rho\rangle) = \langle f(a), \rho\rangle$. *The equivalence relation* $\equiv$ *is defined by the following rule:*

$$\frac{\pi_{\mathcal{A}}(a) = \rho_{\mathcal{A}}(b)}{\left\langle a, \pi^{+1}\right\rangle \equiv \left\langle b, \rho^{+1}\right\rangle}$$

We now provide a definition of the unit and the counit of the adjunction. It suffices to exhibit an universal arrow from $\mathcal{A}$ to $\delta$, whose target is exactly $\delta(F(\mathcal{A}))$, i.e. an arrow $\eta_{\mathcal{A}} : \mathcal{A} \to \delta(F(\mathcal{A}))$ such that, for each $f : \mathcal{A} \to \delta(\mathcal{B})$ , there exists an unique $f^{\#}$ making the diagram in figure 1 commute.

**Theorem 3.5** *An universal arrow* $\eta_{\mathcal{A}} : \mathcal{A} \to \delta(F(\mathcal{A}))$ *is defined* as $\eta_{\mathcal{A}}(a) = \langle a, id\rangle$. *For each* $f : \mathcal{A} \to \delta(\mathcal{B})$, *we have* $f^{\#} : F(\mathcal{A}) \to \mathcal{B}$, with $f^{\#}(\langle a, \rho\rangle) = \rho_{\mathcal{B}}(f(a))$.

The usual definition of adjunction relies on an isomorphism of homsets, which obtains for each arrow $f : \mathcal{A} \to \delta(\mathcal{B})$ simply the arrow $f^{\#} : F(\mathcal{A}) \to \mathcal{B}$. As it is well known from the theory of adjunctions (see e.g. [23]), the universal arrow $\eta_{\mathcal{A}}$, defined for each object $\mathcal{A}$ of $\mathbf{Alg}^{\pi}$, can be seen as a natural transformation $\eta : Id \to \delta \circ F$, which is the unit of the adjunction. The counit is given by $\epsilon : F \circ \delta \to Id$, such that $\epsilon_{\mathcal{A}} = id^{\#}_{\delta(\mathcal{A})}$. By expanding the definition, we obtain the following:

**Definition 3.6** *The counit* $\epsilon_{\mathcal{A}} : F(\delta(\mathcal{A})) \to \mathcal{A}$ *of the adjunction between* $\delta$ *and* $F$ *is defined, for each permutation algebra* $\mathcal{A}$, *as* $\epsilon_{\mathcal{A}}(\langle a, \rho\rangle) = \rho_{\mathcal{A}}(a)$.

# 4 Properties of Abstraction and Concretion

In this section we study the support, symmetry and orbits of elements of finitely supported permutation algebras obtained using $\delta$ and $F$. In particular, we show that both functors restrict from $\mathbf{Alg}^{\pi}$ to $\mathbf{FSAlg}^{\pi}$, and how in $\delta(\mathcal{A})$ we find more distinct orbits than in $\mathcal{A}$, containing the *hidden* elements of $\delta(\mathcal{A})$, i.e. those that have a "bound" name, which is not observable. In $F(\mathcal{A})$ instead, we find a name

*freely* added to the support of each element of $\mathcal{A}$, and an isomorphic set of orbits. The counit plays exactly the role to "reveal" a bound name, which is a non-trivial operation due to lemma 2.12. In section 4.3 we compare our notion of abstraction with the one defined by Gabbay and Pitts, showing that the difference is the same that exists between the ordinary axioms of $\alpha$-conversion and the solution to the problem of binding given by *De Bruijn indexes*. For the purpose, we describe the syntax of the $\lambda$-calculus as an initial algebra for a functor employing $\delta$.

### *4.1 Properties of Abstraction*

**Theorem 4.1** *The support and symmetry of elements of $\delta(\mathcal{A})$ are obtained as $supp_{\delta(\mathcal{A})}(a) = \{i-1 \mid i \in supp_{\mathcal{A}}(a) \setminus 0\}$, and $\mathcal{G}_{\delta(\mathcal{A})}(a) = \{\pi \mid \pi^{+1} \in \mathcal{G}_{\mathcal{A}}(a)\}$.*

The above theorem proves that $\delta$ restricts from $\mathbf{Alg}^{\pi}$ to $\mathbf{FSAlg}^{\pi}$. The intuition behind it is that, in $\delta(\mathcal{A})$, we remove 0 from the support of each element. This way, name $0 \in supp_{\mathcal{A}}(a)$ becomes *fresh* in $\delta(\mathcal{A})$: no observation can be made about it, but it is still a hidden name of $a$. This name can be used, exploiting the action of $\delta$ on arrows: we just have $\delta(f)(a) = f(a)$, hence $f$ can use all the names of $a$.

The property of 0 being *fresh* is also assured by the symmetry of $a$ in $\delta_{\mathcal{A}}$: $\mathcal{G}_{\delta(\mathcal{A})}(a)$ is the subgroup of $\mathcal{G}_{\mathcal{A}}(a)$ that fixes 0, shifted by one name. Information about interchangeability of 0 is thrown away, making it distinct from any other name.

We now define a set of permutations used to describe orbits of $\delta(\mathcal{A})$. Below, the finite set $S$ will be used as the support of an element of a permutation algebra, hence by the convention of remark 2.10 we define these permutations only on $S$.

**Definition 4.2** *Given a finite set $S$, we define a permutation $\pi^{old(S)}$* such that *$\pi^{old(S)}(i) = i+1$ for $i \in S$, and $|S|$ permutations $\pi^{h(S,i)}$, for $i \in S$, such that $\pi^{h(S,i)}(i) = 0$, and $\pi^{h(S,i)}(j) = j+1$ if $j \in S \setminus i$.*

Now we define functions in **Set** acting on carriers of permutation algebras. One is called *old*, because it embeds an element $a$ from $\mathcal{A}$ into $\delta(\mathcal{A})$ preserving all of its properties (support, symmetry, orbit). The other ones are called *hidden* since they obtain, from $a$, new elements in $\delta(\mathcal{A})$, whose properties can not be recovered in $\mathcal{A}$.

**Definition 4.3** *The* old *element $old_{\mathcal{A}}(a)$ and the $i^{th}$* hidden *element $hid^i_{\mathcal{A}}(a)$ of $a \in A$ are defined as $old_{\mathcal{A}}(a) = \pi_{\mathcal{A}}^{old(supp_{\mathcal{A}}(a))}(a)$, and $hid^i_{\mathcal{A}}(a) = \pi_{\mathcal{A}}^{h(supp_{\mathcal{A}}(a),i)}(a)$.*

It is straightforward to check that *old* is a morphism of type $\mathcal{A} \to \delta(\mathcal{A})$. It holds that $supp_{\delta(\mathcal{A})}(old_{\mathcal{A}}(a)) = supp_{\mathcal{A}}(a)$, $\mathcal{G}_{\delta(\mathcal{A})}(old_{\mathcal{A}}(a)) = \mathcal{G}_{\mathcal{A}}(a)$, and $orb_{\delta(\mathcal{A})}(old_{\mathcal{A}}(a)) = \left\{ old_{\mathcal{A}}(x) \,\middle|\, x \in orb_{\mathcal{A}}(a) \right\}$. In other words, *old* is an embedding of $\mathcal{A}$ in $\delta(\mathcal{A})$.

The crucial property of $hid^i$ is to send name $i$ to 0, hence we have (by theorem 4.1), $supp_{\delta(\mathcal{A})}\left(hid^i_{\mathcal{A}}(a)\right) = supp_{\mathcal{A}}(a) \setminus i$. In words, for each element $a$ and each name $i$ of $a$, we can identify an element of $\delta(A)$ which has the same names as $a$, minus $i$. As we will see, such an operation is of fundamental importance to define coalgebras for $\delta$, allowing these to hide names along transitions.

**Remark 4.4** *In the following, we will use $hid^i_{\mathcal{A}}(a)$ to define an element of $\delta(\mathcal{A})$: the subscript $\mathcal{A}$ denotes the application of the permutation action in $\mathcal{A}$, thus identifying an element of the carrier $A$, which is also the carrier of $\delta(\mathcal{A})$, not the fact that element $hid^i_{\mathcal{A}}(a)$ belongs to permutation algebra $\mathcal{A}$ as one might expect.*

We show that the old and hidden elements form a partition of a permutation algebra.

**Lemma 4.5** *For each $a \in A$, there exist either $b \in A$ such that $a = old_{\mathcal{A}}(b)$, or $b \in A$ and $i \in supp(b)$ such that $a = hid^i_{\mathcal{A}}(b)$.*

Using this basic meta-language, we can relate orbits of $\delta(\mathcal{A})$ to orbits of $\mathcal{A}$. For each orbit in $\mathcal{A}$, represented by $a^o_{\mathcal{A}}$, there is a corresponding orbit in $\delta(\mathcal{A})$ without any hidden name, plus as many orbits in $\delta(\mathcal{A})$ as the possible abstractions of names in $supp_{\mathcal{A}}(a^o_{\mathcal{A}})$, modulo its symmetry: there are as many ways to hide a name in $a^o_{\mathcal{A}}$ as names in its support, up-to an equivalence relation saying that there is no difference in abstracting two names, when they are swapped by some permutations in $\mathcal{G}_{\mathcal{A}}(a^o_{\mathcal{A}})$.

**Theorem 4.6** *For $a \in A$, let $H^a = \{old_{\mathcal{A}}(a)\} \cup \{hid^i_{\mathcal{A}}(a) \mid i \in supp_{\mathcal{A}}(a)/_{\equiv}\}$, with $i \equiv j \iff \exists \pi \in \mathcal{G}_{\mathcal{A}}(a).\pi(i) = j$. Let $A^o_{\mathcal{A}}$ be a set of canonical representatives for $\mathcal{A}$. A set $A^o_{\delta(\mathcal{A})}$ for $\delta(\mathcal{A})$ is obtained as $A^o_{\delta(\mathcal{A})} = \bigcup_{a \in A^o_{\mathcal{A}}} H^a$*

The result on orbits also asserts a fundamental property: *hidden* elements can never be turned into *old* elements employing the permutation action of $\delta(\mathcal{A})$, hence they actually are new elements in the resulting algebra. The following example illustrates the need for an existential quantification over $\mathcal{G}_{\mathcal{A}}(a)$.

**Example 4.7** *Applying the result of theorem 4.6 to symmetries obtained by* round shifts *may look counterintuitive. Consider the set of $\pi$-calculus agents with names in $\omega$, up to structural equivalence, seen as a permutation algebra $Pi = \langle A_{Pi}, \{\pi_{Pi}\}\rangle$, and agent $P(1,2,3) = \bar{1}2 + \bar{2}3 + \bar{3}1$. Its symmetry is $\{id, \sigma, \sigma^2\}$, generated by the round shift $\sigma(1) = 2, \sigma(2) = 3, \sigma(3) = 1$. The three agents $P_1 = (\nu 1)P(1,2,3)$, $P_2 = (\nu 2)P(1,2,3)$ and $P_3 = (\nu 3)P(1,2,3)$ belong to the same orbit due to structural equivalence. However, the symmetry of $P_1$ (and consequently, of $P_2$ and $P_3$ which are on the same orbit) is just $\{id\}$: the support of $P_1$ is $\{2,3\}$, hence the only possible candidate permutation besides the identity is the swap $\rho(2) = 3, \rho(3) = 2$, but $\rho_{Pi}(P_1) = (\nu 1)(\bar{1}3 + \bar{3}2 + \bar{2}1)$ which is not structurally equivalent to $P_1$ itself, whereas one might have expected $\rho \in \mathcal{G}_{Pi}(P_1)$.*

Finally, we observe that, being a right adjoint, $\delta$ admits a final coalgebra: $\mathbf{FSAlg}^{\pi}$ has a final object 1 where each permutation acts as the identity. Since right adjoints preserve final objects, the final coalgebra of $\delta$ is just $id : 1 \to 1$.

### *4.2 Properties of Concretion*

The following theorem shows that each element of $F(\mathcal{A})$ has an additional name, which is added "syntactically" or "freely", i.e. is not obtained by properties of $\mathcal{A}$.

**Theorem 4.8** *We have $supp_{F(\mathcal{A})}(\langle a, \rho\rangle) = \{\rho(0)\} \cup \{\rho(i+1) \mid i \in supp_{\mathcal{A}}(a)\}$.*

The symmetry of an element of $F(\mathcal{A})$ is given by the symmetry of $a$ in $\mathcal{A}$, translated using the right shift theory morphism and the permutation $\rho$.

**Theorem 4.9** *The symmetry $\mathcal{G}_{F(\mathcal{A})}(\langle a, \rho\rangle)$ is given by $\{\rho \circ \pi^{+1} \circ \rho^{-1} \mid \pi \in \mathcal{G}_{\mathcal{A}}(a)\}$.*

Notice that no permutation in the symmetry can swap (shifted) names of $a$ and 0, hence 0 is *distinguished* from names already in $a$.

We finally analyze the set of orbits of $F(\mathcal{A})$, which is isomorphic to that of $\mathcal{A}$.

**Theorem 4.10** *Given a permutation algebra $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$, a set of canonical*

$$f(\lambda x.l) = \langle 0, hid^x_\Lambda(f(l))\rangle \text{ if } x \in fn(l) \qquad f(l_1 l_2) = \langle 1, \langle f(l_1), f(l_2)\rangle\rangle$$
$$f(\lambda x.l) = \langle 0, old_\Lambda(f(l))\rangle \text{ if } x \notin fn(l) \qquad f(x) = \langle 2, x\rangle$$

Figure 2.

*representatives of orbits of* $F(\mathcal{A})$ *is given by* $F(\mathcal{A})^o = \{\langle a, id\rangle \mid a \in A^o_{\mathcal{A}}\}$.

For each $a \in A$ we can recover, as we did for abstraction (employing *old*), an element having the same properties of $a$ in $\mathcal{A}$, plus the addition of a new name.

**Lemma 4.11** *For each element* $a$*, and each name* $i \notin supp_{\mathcal{A}}(a)$ *there exists an element of* $F(\mathcal{A})$ *whose support is* $supp_{\mathcal{A}}(a) \cup \{i\}$*, with* $\mathcal{G}_{F(\mathcal{A})}(a) = \mathcal{G}_{\mathcal{A}}(a) \cap fix(i)$.

The contents of this section, and in particular the above lemma, amount to say that the algebra $F(\mathcal{A})$ is, by theorem 2.13, isomorphic to the algebra $\mathcal{FA} = \langle FA, \{\pi_{\mathcal{FA}}\}\rangle$ where $FA = \{\langle n, a\rangle \in \omega \times A \mid n \notin supp_{\mathcal{A}}(a)\}$, and $\pi_{\mathcal{FA}}$ is the permutation action over the product algebra $\omega \times \mathcal{A}$, which is $\pi_{\mathcal{FA}}(\langle n, a\rangle) = \langle \pi(n), \pi_{\mathcal{A}}(a)\rangle$.

### *4.3 Comparison with Abstraction and Concretion in Nominal Sets*

Abstraction $[i]a$ for an element $a$ of a nominal set $\mathcal{A}$ and a name $i$ (see [7]) is defined as the equivalence class obtained by swapping $i$ with a name $j$ in the pair $\langle i, a\rangle$, for all possible names $j$ not in the support of $a$. This is quite the idea of representing the axioms of $\alpha$ -conversion, while the idea of shifting names, and calling the bound name 0, is typical of the *De Bruijn indexes* approach [4]. Nevertheless, objects obtained from the two constructions are isomorphic by theorem 2.13.

On the other hand, the *freshness relation* $i\#a$ in nominal syntax is defined as the set of elements $a$ paired with names $i$ which are *fresh* for $a$, i.e. not in its support. This is a nominal set when equipped with the ordinary action of the permutation on the product. Observing the conclusion of section 4.2, the nominal set of freshness is isomorphic to the object $F(\mathcal{A})$. Hence, the adjunction associated to the definition of the right shift theory morphism provides a proof of the adjunction between abstraction and concretion in the setting of nominal sets.

To illustrate how De Bruijn indexes are obtained using $\delta$, we show an encoding of the syntax of the $\lambda$-calculus. We assume finite products and coproducts to be defined in $\mathbf{FSAlg}^\pi$ (these functors are trivially lifted from **Set**, defining the permutation operation pointwise), and, given a finite coproduct $X_0 + X_1 + \ldots + X_n$ we denote its elements with $\left\{\langle i, x\rangle \,\middle|\, i \in \{0, \ldots, n\} \wedge x \in X_i\right\}$.

**Example 4.12** (De Bruijn indexes) *Consider the set of* $\lambda$*-calculus terms, defined by the syntax* $L ::= \lambda x.L \mid LL \mid x$*, for* $x \in \omega$*. Instead of introducing the notion of* $\alpha$*-equivalence for terms, we can define the syntax as the term algebra* $\Lambda$ *of the functor* $T$*, i.e. as an arrow in* $\mathbf{FSAlg}^\pi$ *of type* $T(X) \to X$*, where* $T(X) = \delta(X) + X \times X + \omega$*, and* $\omega$ *is seen as a permutation algebra with* $\pi_\omega(i) = \pi(i)$*. Notice that, being* $T$ *a functor in* $\mathbf{FSAlg}^\pi$*, an action of the permutation* $\pi_\Lambda$ *on* $\Lambda$ *is defined by initiality using* $\pi_\omega$ *as the base case, thus introducing support, symmetry and orbits of elements. An interpretation* $f$ *of* $\lambda$*-terms as elements of* $\Lambda$ *is given in figure 2.*

*The most important case is the usage of* $hid^x_\Lambda$ *to define* $f(\lambda x.l)$*: it shifts all names of* $l$ *by one, and assigns the name* 0 *to the bound name. There is more: all*

*the $\alpha$-equivalent terms of the form $\lambda y.l\left[{}^{y}/_{x}\right]$ are translated into the same element $t$ of $\Lambda$, where name 0 is not "visible" in $supp_{\Lambda}(t)$, and permutations can not exchange it with any other. This corresponds to a notion of "capture-avoiding permutation" for $\Lambda$. In fact, by theorem 4.1, we have $supp_{\Lambda}(t) = supp_{\lambda}(f(t)) \setminus \{x\}$.*

**Remark 4.13** *As a side result of this work, we observe that the* concretion operation *of Gabbay and Pitts, which closely resembles the* reveal *connective of spatial logics such as [19,20], arises as the counit of the adjunction. The action of the counit is $\epsilon_{\mathcal{A}}(\langle a, \rho\rangle) = \rho_{\mathcal{A}}(a)$. Notice that $\rho$ is also applied to name 0, differently from the action of $\rho$ in $\delta(\mathcal{A})$. Let $\rho(i) = 0, \rho(0) = i$, with $i \notin supp_{\mathcal{A}}(a)$. We have $\epsilon_{\mathcal{A}}(\langle a, \rho\rangle) = a\left[{}^{i}/_{0}\right]$: the hidden name 0 is "bound" by $\rho$ to the fresh $i$, using a morphism of the category. This operation is not trivial: names of $\rho_{\mathcal{A}}(a)$ are possibly more than names of $a \in \delta(\mathcal{A})$, hence by lemma 2.12 such an operation must have, as a domain, at least $F(\delta(\mathcal{A}))$. Even though it is not a major concern in this work, and the topic should be developed in detail, finding a place in the diagram for a* reveal *operation clarifies the formal relationship between the dual notions of abstraction and freshness, and that of* revelation. *This relationship can easily be mistaken, since these notions have often been jointly used.*

# 5 Coalgebras over Nominal Sets for the Semantics of Name Passing Calculi

In this section, we show how $\delta$ can be used to express the early semantics of the $\pi$-calculus as a coalgebra in $\mathbf{FSAlg}^{\pi}$, hence in nominal sets. We closely follow [12], where the early semantics of the $\pi$-calculus is given in the form of *Structural Operational Semantics* rules, implicitly defining a functor for the $\pi$-calculus. Results in [16] ensure a *lifting* of the semantics from the category **Set** to the category $\mathbf{Alg}^{\pi}$ that respects axioms, obtaining a bialgebra where the algebraic operations and axioms are those of the permutation signature.

Our contribution here is to give an explicit form for the functor employed in [12], which is $T(X) = \mathcal{P}_{fs}(\mathcal{L}' \times X + \mathcal{L}'' \times \delta(X))$, where $\mathcal{L}'$ and $\mathcal{L}''$ are disjoint, and $\mathcal{P}_{fs}$ is a finitely-supported variant of the power set functor. By posing $\mathcal{L} = \mathcal{L}' \cup \mathcal{L}''$, and using the embedding of $X$ into $\delta(X)$, one could just define $T(X) = \mathcal{P}_{fs}(\mathcal{L} \times \delta(X))$. Notice how the latter is much closer to ordinary labeled transition systems in **Set**, than many definitions of endofunctors in presheaf categories. A possible reason is that in $\mathbf{Set}^{\mathbf{I}}$ each process is given a "type" representing its support, thus the semantics carries some redundant information that is missing in its untyped counterpart, nominal sets, and can be discarded without loosing full abstractness.

The contribution of this section might thus be summarized as showing that the category of nominal sets, together with the abstraction functor, allows one to enrich in a natural way ordinary labeled transition systems with operators for dynamic allocation of names.

## *5.1 Hiding for the early semantics of the $\pi$-calculus*

For $\langle A, \{\pi_{\mathcal{A}}\}\rangle$ permutation algebra, and $S \in \mathcal{P}(A)$, let the action of a permutation on any subset of $A$ be defined as $\pi_{\mathcal{P}(\mathcal{A})}(S) = \{\pi_{\mathcal{A}}(a) \mid a \in S\}$, extended to a

$$\text{rule 1: } \frac{X \overset{l}{\Longrightarrow} X'}{\rho(X) \overset{\rho(l)}{\Longrightarrow} \rho(X')} \text{ for } l \in \mathcal{L}' \qquad \text{rule 2: } \frac{X \overset{bout(x)}{\Longrightarrow} X'}{\rho(X) \overset{\rho(bout(x))}{\Longrightarrow} \rho^{+1}(X')}$$

Figure 3.

$$\frac{p \overset{\tau}{\longrightarrow} p'}{p \overset{tau}{\longrightarrow}_f p'} \qquad \frac{p \overset{xy}{\longrightarrow} p'}{p \overset{in(x,y)}{\longrightarrow}_f p'} \qquad \frac{p \overset{\bar{x}y}{\longrightarrow} p'}{p \overset{out(x,y)}{\longrightarrow}_f p'} \qquad \frac{p \overset{\bar{x}(y)}{\longrightarrow} p'}{p \overset{bout(x)}{\longrightarrow}_f hid^y_{Pi}(p')}$$

Figure 4.

*power set* endofunctor in $\mathbf{Alg}^\pi$. The *countable power set* functor $\mathcal{P}_\omega$ can then be defined as in **Set**. However, it does not restrict to an endofunctor in $\mathbf{FSAlg}^\pi$. For example, the set of odd numbers is not finitely supported as an element of $\mathcal{P}_\omega(\omega)$. This motivates the definition of a *countable finitely supported power set*, which we denote with $\mathcal{P}_{fs}$. Elements of $\mathcal{P}_{fs}(\mathcal{A})$ are all the elements of $\mathcal{P}_\omega(\mathcal{A})$ having the finite support property.

The definition of $\mathcal{P}_{fs}$ actually addresses a potential question that might arise on the results in [12]: the lifting of the semantics is to $\mathbf{Alg}^\pi$, not $\mathbf{FSAlg}^\pi$. However, the theorem is applied to ensure that a final coalgebra semantics of the early $\pi$-calculus exists as a finitely supported permutation algebra. The reason why this holds is that $\mathbf{FSAlg}^\pi$ is a *full* subcategory of $\mathbf{Alg}^\pi$, hence the image of each morphism yields a finitely supported algebra out of any finitely supported algebra.

In the following, we define two permutation algebras of labels, with $x$ and $y$ ranging over $\omega$, having syntactic substitution as the permutation action $\pi_{\mathcal{L}}$: $\mathcal{L}'$ with carrier $\{tau, in(x,y), out(x,y)\}$ and $\mathcal{L}''$ with carrier $\{bout(x)\}$.

A *transition specification* [16] is a set of "meta-rules" that specify the possible format of transition rules, allowing the transition system to lift from **Set** to an appropriate category of algebras. The transition specification $\Delta_\pi$ for the $\pi$-calculus given in [12] can be found in figure 3. These rules describe the action of algebraic operations on transitions, obtaining a bialgebra in $\mathbf{Alg}^\pi$.

**Theorem 5.1** *Let $T(X) = \mathcal{P}_{fs}(\mathcal{L}' \times X + \mathcal{L}'' \times \delta(X))$. A transition function in $\mathbf{FSAlg}^\pi$ is a coalgebra for $T$ if and only if it respects $\Delta_\pi$.*

This shows that $\delta$ is essentially the only addition needed to represent the early semantics of the $\pi$-calculus in a purely coalgebraic way. In $\mathbf{FSAlg}^\pi$, bisimulation for the early $\pi$-calculus is the coalgebraic notion, without any side condition on bound names, due to proper handling of fresh names.

Let $Pi$ denote the permutation algebra of $\pi$-calculus agents whose permutation action is defined as the standard notion of substitution. An arrow $f : Pi \to T(Pi)$ representing the semantics of the $\pi$-calculus is described by the rules in figure 4. As in [12], we employ the transition system in **Set** in premises of the rules. Notice that permutation $\sigma$, applied to $p'$ in rule for bound output in [12], corresponds exactly to $hid^y_{Pi}(p')$, hence the coalgebra provided there is the same as the one we introduced.

# 6 Named Sets and HD-Automata

Named sets with symmetries arise from permutation algebras, after observing the effectiveness of this model to faithfully represent the semantics of name-passing calculi, but the failure to re-use names, that is, to model garbage collection. Named sets are a computationally efficient representation of permutation algebras (thus, of nominal sets and Schanuel topos), allowing systems represented as coalgebras in this category to be minimized and verified. The theory of named sets has been developed in various stages, however it can be captured, *a posteriori*, by two basic facts.

The first distinguishing feature of the category is *locality* of names: the meaning of a name is not assumed to be globally known, but rather relationships between names of different elements are established locally by morphisms of the category. For this reason, elements that are on the same orbit can be safely identified. This way, garbage collection is modeled. Consider for example the $\pi$-calculus agent $P(a) = (\nu x)\,\bar{a}x.P(x)$. Due to freshness of names, the standard LTS semantics can reach all the states in the (infinite) orbit $\left\{ P(x) \,\middle|\, x \in \omega \right\}$. Identifying all elements of an orbit, we just have one state in the system, $P(a)$. The transition function has an associated "relocation" function for names, that at each step discards names that are no longer used, and at the same time can allocate a *new* one.

The second novel aspect of named sets is based on the observation that the essence of permutation algebras lies in the *symmetry* of *orbits*. Two objects that have isomorphic sets of orbits, and for each pair of orbits in the isomorphism, a pair of elements with the same symmetry, are isomorphic in turn as objects of the category, as shown in theorem 2.13.

From this fact, and theorem 2.14, comes the representation of named sets: they are sets whose elements have an attached group of permutations, acting, for each element, on a finite set of names. Arrows of the category reflect this structure: they are total functions, equipped with injective renamings that "go backwards" and represent *history* of names: hence the name of *history-dependent automata* for the coalgebras of this category.

## *6.1 Named Sets*

Here we present the category **NSet**, as it was defined in [17]. Similarly to [14,15], we try to adopt a simpler notation, in particular omitting an ordering over elements, which is not needed here. We denote with $\prod_{x \in S} t(x)$ the set-theoretical *dependent product* construction, i.e. the function space $x \to t(x)$ for $x$ ranging over $S$ and $t(x)$ a set, and with $Aut\,(\mathcal{N})$ the set of all permutations over the finite set $\mathcal{N}$.

**Definition 6.1** *(named set) A* named set $N$ *is a pair, consisting of a set* $Q_N$ *and, for each element* $q$ *of* $Q_N$*, a group of permutations* $\mathcal{S}_N(q)$ *over a finite set* $\mathcal{N}_q \subset \omega$*:*

$$N = \langle Q_N, \mathcal{S}_N : \prod_{q \in Q_N} Aut\,(\mathcal{N}_q) \rangle$$

We can recover the *support* $\|-\|_N : Q_N \to \mathcal{P}_{fin}(\omega)$ as $\|q\|_N = dom\,(\mathcal{S}_N(q))$. The intuition is that each element of $Q_N$ is the canonical representative of an orbit of a permutation algebra, up-to algebra isomorphism. The restriction of the symmetry

of $q \in Q_N$ to its support is described by $\mathcal{S}_N(q)$. In virtue of theorem 2.14, it is not necessary to describe the action of the symmetry on names outside the support.

**Definition 6.2** *(named function) A* named function $F : N \to M$ *between named sets $N$ and $M$ is made up of a function $h_F$, and for each element $q$ of $q_N$ a set of injective functions $\Sigma_F(q)$, each one with type $\|h_F(q)\| \overset{inj}{\to} \|q\|$*

$$F = \langle h_F : Q_N \to Q_M, \Sigma_F : \prod_{q \in Q_N} \mathcal{P}\left(\|h_F(q)\|_M \overset{inj}{\to} \|q\|_N\right)\rangle$$

*subject to the following additional constraints, for each $q \in Q_N$, and $\sigma \in \Sigma_F(q)$:*

(1) $\sigma \circ \mathcal{S}_M(h_F(q)) = \Sigma_F(q)$

(2) $\mathcal{S}_N(q) \circ \sigma \subseteq \Sigma_F(q)$

Condition (1) is equivalent to say that $\Sigma_F$ is given by $\sigma \circ \mathcal{S}_M(q)$ for $\sigma : \|(q)\|_M \overset{inj}{\to} \omega$. The intuition is that $\Sigma_F$ is a name mapping, tracing the history of names of $q$ when mapped via $h_F$. However, $\Sigma_F$ cannot be just the injective function $\sigma$, because that would distinguish names that are in the symmetry of $h_F(q)$. Thus, $\Sigma_F$ is an injective function, saturated by composition with $\mathcal{S}_M(q)$.

Condition (2) has not been given a particularly intuitive meaning in previous work on named sets. We try here to explain its purpose: by substitution of (1) in (2) we obtain $\mathcal{S}_N(q) \circ \sigma \subseteq \sigma \circ \mathcal{S}_M(h_f(q))$, then $\mathcal{S}_N(q)_{|_{cod(\sigma)}} \subseteq \sigma \circ \mathcal{S}_M(h_f(q)) \circ \sigma^{-1}$. This is the same as thesis of theorem 2.11, asserting that symmetry grows along morphisms: violating this condition would result in named functions that do not represent permutation algebra morphisms. Due to locality of names, the symmetry $\mathcal{S}_M$ has to be translated to the domain of $\mathcal{S}_N$ exploiting the name mappings $\sigma \in \Sigma$ and $\sigma^{-1}$, in order to allow the two to be compared.

**Definition 6.3** *(identity and composition) The* identity *named function is $id_N = \langle id_{Q_N}, \lambda q.\mathcal{S}_N(q)\rangle$. The* composition *of two named functions $F : N \to M$ and $G : M \to O$ is given by $G \circ F = \langle h_G \circ h_F, \Sigma\rangle$, where $\Sigma(q) = \Sigma_F(q) \circ \Sigma_G(h_f(q))$.*

The proof that composition of named functions yields a named function, hence that named sets and named functions form a category, can be found in [18].

A further explanation can be given for names as modeling a number of *local* resources attached e.g. to a state of a system, and for symmetries as denoting indistinguishability of some resources. The backward set of name mappings in a named function traces the history of resources along the function, in particular identifying those that are preserved, and those that are discarded.

We recall the following theorem, proved as proposition 29 in [14].

**Theorem 6.4** *The categories* **NSet** *and* $\mathbf{FSAlg}^{\pi}$ *are* equivalent*: there exist two functors $E : \mathbf{NSet} \to \mathbf{FSAlg}^{\pi}$ and $G : \mathbf{FSAlg}^{\pi} \to \mathbf{NSet}$ such that the compositions $G \circ E$ and $E \circ G$ are isomorphic to the two identity functors.*

The meaning of the above theorem is that there exist constructions to obtain a permutation algebra from a named set, and conversely a named set from a permutation algebra, in such a way that going in one direction, and then back, gives an isomorphic object. We introduce the functor $G$, used in theorem 7.3.

**Definition 6.5** *$G$ sends $\mathcal{A} = \langle A, \{\pi_{\mathcal{A}}\}\rangle$ to $N^{\mathcal{A}} = \langle A^o, \mathcal{S}_N\rangle$, where $\mathcal{S}_N(a^o) =$*

$\mathcal{G}_{\mathcal{A}}(a^o)_{|_{supp_{\mathcal{A}}(a^o)}}$. *For* $f : \mathcal{A} \to \mathcal{B}$, *let* $\rho$ *be such that* $\rho_{\mathcal{B}}(f(a^o)^o) = f(a^o)$. $G$ *sends* $f$ *to* $F = \langle h_F, \Sigma_F \rangle$, *where* $h_F(a^o) = f(a^o)^o$, *and* $\Sigma_F(a^o) = \rho \circ \mathcal{S}_N(f(a^o)^o)$.

Notice that composition with the symmetry of $f(a^o)^o$ restricts the permutation $\rho$, (whose inverse normalizes $f(a^o)$) to $supp_{\mathcal{B}}(f(a^o)^o) = \|f(a^o)^o\|_{G(\mathcal{B})}$, turning it into a bijection from this set to $supp_{\mathcal{B}}(f(a^o)) \subseteq supp_{\mathcal{A}}(a^o) = \|a^o\|_{G(\mathcal{A})}$, as required by definition 6.2. We also recall the definition of $E$, to explain locality of names.

**Definition 6.6** *The functor* $E$ *acts on each object* $N$, *returning* $\mathcal{A}^N = \langle A_N, \{\pi_N\} \rangle$, *where* $A_N = \{\langle q, \rho \circ \mathcal{S}_N(q) \mid q \in Q_N, \rho \in Autf \rangle\}$, *and* $\pi_N(\langle q, I \rangle) = \langle q, \pi \circ I \rangle$. $E$ *acts on each arrow* $F$ *returning the morphism* $f(\langle q, I \rangle) = \langle h_F(q), I \circ \Sigma_F(q) \rangle$.

The point is that, when mapping back a named set to a permutation algebra, the whole orbit that $q \in Q_N$ represents is reconstructed by the permutation action $\pi_N$. Thus, one has the choice of a permutation $\rho$ which maps the *local* names $\|q\|_N$ of $q$ to $\omega$, giving a *global* meaning to them: actually, employing the above definition, it is easy to see that $supp_{\mathcal{A}^N}(\langle q, \rho \circ \mathcal{S}_N(q) \rangle) = \rho(\|q\|_N)$. Composition of the permutation $\rho$ with $\mathcal{S}_N(q)$ ensures that the obtained permutation action $\pi_N$ respects the symmetry of $q$ itself. A morphism follows the mapping $\rho$, by just "carrying it on" using the history of names $\Sigma_F$: we have $f(\langle q, \rho \circ \mathcal{S}_N(q) \rangle) = \langle h_F(q), \rho \circ \mathcal{S}_N(q) \circ \Sigma_F(q) \rangle$. The initial choice of $\rho$ uniquely determines the mapping of local names in elements of the destination of $F$.

**Remark 6.7** *The necessity to choose a permutation* $\rho$ *expresses precisely the notion of locality of names. As a practical example, consider the addition of a fragment of code to an existing program. The first thing a programmer should do is to rename all free variables of the pasted fragment, in order to avoid name clashes. In doing so, one can as well rename the variables of the whole target program, without affecting the meaning of the composition, provided that clash is avoided. This is the same as stating, in our terminology, that the free variables of a program fragment are local names, and their global meaning has to be established when it is needed.*

# 7 Abstraction for Named Sets and Garbage Collection

In this section we define the endofunctor of abstraction in named sets, which we call $H$ (standing for "hiding"). In section 7.1, we give the formal definition, and a theorem that establishes a correspondence with the definition in $\mathbf{FSAlg}^{\pi}$, exploiting the equivalence result given in [14]. Then, in section 7.2, we give an account on how unused names are discarded, by means of some example, to the aim of motivating the main "slogan" we would like to propose in this work: *named sets are nominal sets plus garbage collection.*

## *7.1 The Abstraction Functor in* **NSet**

Abstraction can be defined in named sets with the aid of theorems 4.1 and 4.6.

The underlying set $Q_{H(N)}$ resulting from the action of $H$ on an object $N$ is the union of $Q_N$ itself, representing the orbits of the *old* elements of definition 4.3, and a set of pairs $\langle q \in Q_N, i \in \|q\|_N \rangle$, representing the orbit of the $i^{th}$ hidden element. Intuitively, $i$ marks the $i^{th}$ name of $q$ as hidden. As we did in theorem 4.6,

the possible values for $i$ have to be quotiented using the symmetry of $q$. We pose $i \equiv_q j \iff \exists \pi \in \mathcal{S}_N(q).\pi(i) = j$, and define

$$Q_{H(N)} = Q_N \cup \left\{ \langle q, i \rangle \middle| q \in Q_N, i \in (\|q\|_N)/_{\equiv_q} \right\}$$

For readability, in all the following definitions, we implicitly assume the pattern matching on $q$ and $\langle q, i \rangle$ to have the additional constraint $q \in Q_N$, to avoid clashes. The symmetry of elements of the form $\langle q, i \rangle$ is defined as the subgroup of the symmetry of $q$ that fixes $i$ (which we denote with $gfix(\mathcal{S}_N(q), i)$) according to theorem 4.1. This symmetry is opportunely restricted in order to exclude $i$ from the support. We thus pose $\mathcal{S}_{H(N)}(q) = \mathcal{S}_N(q)$, and

$$\mathcal{S}_{H(N)}(\langle q, i \rangle) = gfix(\mathcal{S}_N(q), i)_{|\|q\|_N \setminus i}$$

The action of $H$ on arrows maps $F : N \to M$ to $H(F) = \langle h_{H(F)}, \Sigma_{H(F)} \rangle$. A pair $\langle q, i \rangle$ is mapped by $h_{H(F)}$ to a pair $\langle h_F(q), j \rangle$ if and only if $j$ is mapped by some injection in $\Sigma_F(q)$ into $i$, that is, if and only if $i$ is still present in the destination $h_F(q)$, according to the history of names $\Sigma_F(q)$. We have $h_{H(F)}(q) = h_F(q)$, and

$$h_{H(F)}(\langle q, i \rangle) = \begin{cases} \langle h_F(q), j \rangle & \text{if } \exists \sigma \in \Sigma_F(q).\sigma(j) = i \\ h_F(q) & \text{otherwise} \end{cases}$$

Notice that $j$ stands for its canonical representative in $\equiv_{h_F(q)}$. For $h_{H(F)}$ to be well-defined, we have to show that it respects the equivalence relation on hidden names. This comes from condition (1) in the definition of a named function: for each $\sigma \in \Sigma_F(q)$ we have $\Sigma_F(q) = \sigma \circ \mathcal{S}_M(h_F(q))$. If there exist $\sigma' \in \Sigma_F(q)$ and $j' \neq j$ such that $\sigma'(j') = i$, then at least a permutation exchanging $j$ and $j'$ belongs to $\mathcal{S}_M(h_F(q))$, hence $j$ and $j'$ are quotiented by the equivalence relation. The mapping $\Sigma_{H(F)}$ is defined as $\Sigma_{H(F)}(q) = \Sigma_F(q)$, and

$$\Sigma_{H(F)}(\langle q, i \rangle) = \begin{cases} \left\{ \sigma_{|dom(\sigma) \setminus j} \middle| \sigma(j) = i \wedge \sigma \in \Sigma_F(q) \right\} & \text{if } h_{H(F)}(\langle q, i \rangle) = \langle h_F(q), j \rangle \\ \Sigma_F(q) & \text{otherwise} \end{cases}$$

When an hidden name is preserved by $\Sigma_F(q)$ we take the subset of $\Sigma_F(q)$ that sends $j$ into $i$, restricted so that $j$ is not mapped at all. This is the same as taking $\sigma_{|dom(\sigma) \setminus j} \circ G'$, where $G'$ is the subgroup of $\mathcal{S}_M(h_F(q))$ fixing $j$. This follows the correspondence with algebras, and in particular theorem 4.1.

Now we summarize the contents of this section in the following definition.

**Definition 7.1** *The abstraction functor $H$ : **NSet** $\to$ **NSet** is defined on objects as $H(N) = \langle Q_{H(N)}, \mathcal{S}_{H(N)} \rangle$, and on arrows as $H(F) = \langle h_{H(F)}, \Sigma_{H(F)} \rangle$.*

**Theorem 7.2** *$H$ is a functor.*

Finally, we show that $\delta$ and $H$ are related by an isomorphism of functors.

**Theorem 7.3** *The two functors $G \circ \delta$ and $H \circ G$ are isomorphic, i.e. there exists a natural transformation $\iota : G \circ \delta \to H \circ G$ such that each component $\iota_N$ is an isomorphism in* **NSet**.

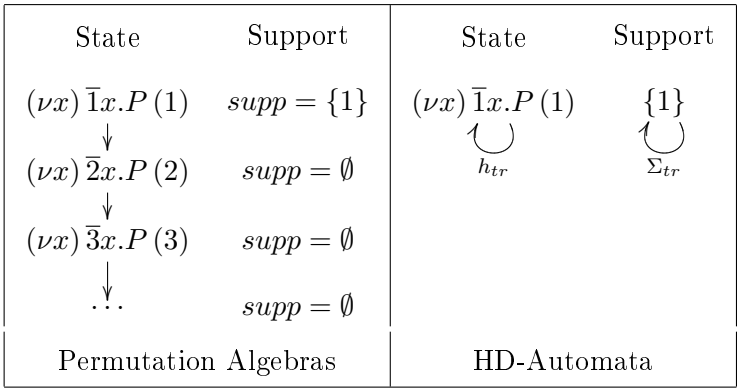


Figure 5.

### *7.2 Garbage Collection*

Consider the definition of $h_{H(F)}\left(\langle q, i\rangle\right)$ in the abstraction functor for named sets and observe that, when the hidden name $i$ is discarded along a morphism, the resulting element is just $q$. This introduces garbage collection, allowing the semantics to reuse old states whenever a fresh name is discarded. We now attempt to give an intuition of this fact by the means of two examples in $\pi$-calculus. We ignore labels of transitions, since these do not contribute to the intuition, and we omit to denote the power set, because both systems are deterministic. When representing HD-automata, we draw the backward mappings of names, together with supports of states, side by side with the transition function. Since the symmetry of both the agents we present is just $\{id\}$, a backward mapping $\Sigma$ is represented by a single function.

Consider the agent $P\,(1) = (\nu x)\,\overline{1}x.P\,(1)$. Even though it has no memory of the past, thus after just one step there are no more free names to be discarded, the permutation algebra semantics of the system reaches all the (infinite, countable) elements in the orbit of[⋆⋆⋆⋆] $P\,(1)$. Figure 5 depicts a sketch of its permutation algebra semantics, compared to its HD-automaton, which is a simple loop. The transition (named) function $tr = \langle h_{tr}, \Sigma_{tr}\rangle$ acts on $P\,(1)$ (intended as the canonical representative of its whole orbit) as $h_{tr}\left(P\,(1)\right) = \{P\,(1)\}$, $\Sigma_{tr}\left(P\,(1)\right) = \{id_{\{1\}}\}$.

Now let $R\,(1) = (\nu x)\,\overline{1}x.R\,(x)$. Consider a presheaf semantics for the $\pi$-calculus. On the left of each state, we draw in figure 5 the least *stage* (object of the base category) in which the state is found at all (the categorial support of the element, as it is called in [14]), and the stage in which the coalgebra is applied to it. Recall that, given a presheaf $T$, the functor $\delta : \mathbf{Set^I} \to \mathbf{Set^I}$ is defined on objects as $\delta(T)\,(X) = T\,(X \oplus 1)$. To distinguish the different instances, in the successive applications of the coproduct, of the only element $* \in 1$ (the final object of $I$), we denote it with $*'$, $*''$ and so on. In the same figure, we can find the HD-automata semantics of $R\,(1)$. The HD-automata semantics is now made up of two states, since in the first step the free name $a$ has to disappear. The transition function is $h_{tr}\left(R\,(1)\right) = \{\langle R\,(1)\,, 1\rangle\}$, thus hiding name 1, and $h_{tr}\left(\langle R\,(1)\,, 1\rangle\right) = \{\langle R\,(1)\,, 1\rangle\}$. We have the name mapping $\Sigma_{tr}\left(R\,(1)\right) = \Sigma_{tr}\left(\langle R\,(1)\,, 1\rangle\right) = \{\emptyset\}$, the empty name mapping. This is required since the support of the destination is empty.

[⋆⋆⋆⋆] So does the nominal sets semantics, even when employing the abstraction functor of Gabbay and Pitts. We omit such an example for space reasons.

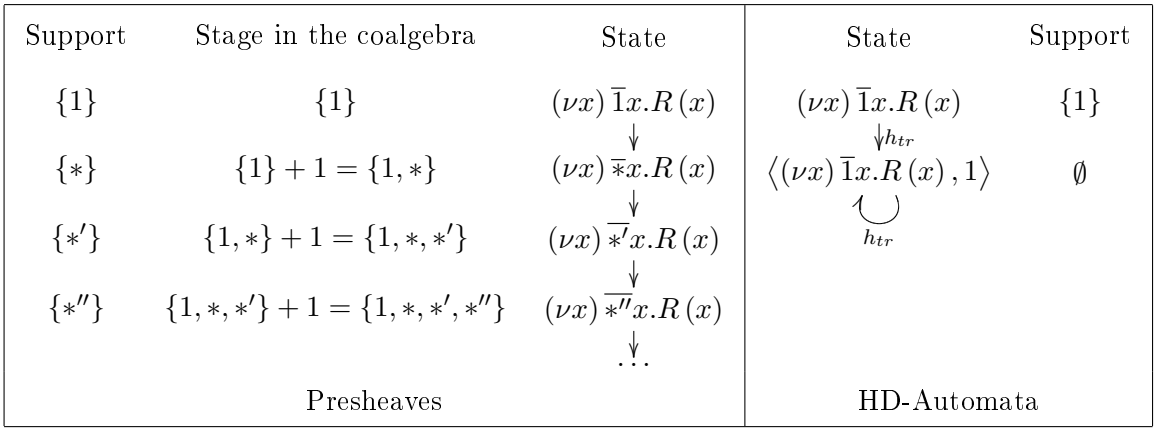


Figure 6.

What both permutation algebras and functors in $\mathbf{Set}^{\mathbf{I}}$ lack, is a mechanism to discard unused names using a quotient operation: on orbits, in the case of permutation algebras, on isomorphic categorial supports, in the case of presheaves.

These examples should explain what we mean with *locality* of names. In particular, notice how the backwards name mappings of named functions trace the history of names along morphisms, allowing the semantics to reuse the same state with different names. One might wonder if this is just a trick and, in the end, one has to perform an "unfolding" of the HD-automata semantics into the ordinary LTS semantics to implement algorithms such as bisimulation checking or model checking. Results in [18] for minimization and bisimulation checking, and work in progress on the model checking side, show that this is not necessary, and the model can be used "as is" to verify systems up-to garbage collection.

## 8 Conclusions and Future Work

We have provided the necessary theory to take in account garbage collection in nominal models of computation, by defining a suitable abstraction functor in the category of named sets. This is essential for the definition of algorithms to handle the syntax and the semantics of name passing calculi. We have also shown that, using the *algebraic* definition of nominal sets, a very simple theory morphism induces a number of commonly used operations in the category of nominal sets: name abstraction, the freshness relation, and the concretion, or reveal, operation.

Future work includes the study of other behavioral functors on named sets, and the proof of a final coalgebra theorem for their composition. The definition of adequate modal logics, and of an efficient model checking algorithm, is yet to be done on history dependent automata *with symmetries* (even though work in progress is promising). In this light, it is useful to define generic functors on named sets, in order to define modal logics using *Stone duality*. The approach proved fruitful to express the logical semantics of nominal calculi [24], and could be applied to coalgebras over named sets to obtain a generic, algorithmic model for calculi with name passing.

In [18], the group-theoretical notion of *generators* is exploited to give a compact representation of elements of named sets in terms of the symmetry of orbits. To improve the minimization algorithm, and in perspective for model checking, effi-

cient group-theoretical algorithms over generators could be employed. We plan to investigate the adoption of algorithms described in [25] to this aim.

Finally, the relationship between the counit of the adjunction, and spatial operators over names such as those in [19,20] should be studied in detail.

# References


[1] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, part i. IC **100**(1) (1992) 1–40

[2] Milner, R., Parrow, J., Walker, D.: Modal logics for mobile processes. TCS **114**(1) (1993) 149–171

[3] Dam, M.: Model checking mobile processes. LNCS **715** (1993) 22–36

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

[5] Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: LICS. (1999) 193–202

[6] Gabbay, M., Pitts, A.: A new approach to abstract syntax involving binders. In: LICS. (1999) 214–224

[7] Gabbay, M., Pitts, A.M.: A new approach to abstract syntax with variable binding. FAC **13**(3-5) (2002) 341–363

[8] Pitts, A.M.: Nominal logic: A first order theory of names and binding. LNCS **2215** (2001) 219–242

[9] Cattani, G.L., Stark, I., Winskel, G.: Presheaf models for the $\pi$-calculus. In: CTCS. (1997) 106–126

[10] Fiore, M.P., Moggi, E., Sangiorgi, D.: A fully abstract model for the $\pi$-calculus. IC **179**(1) (2002) 76–117

[11] Montanari, U., Pistore, M.: *pi*-calculus, structured coalgebras, and minimal hd-automata. In: MFCS. (2000) 569–578

[12] Montanari, U., Pistore, M.: Structured coalgebras and minimal hd-automata for the $\pi$-calculus. TCS **340** (2005) 539–576

[13] Pistore, M.: History Dependent Automata. PhD thesis, Università di Pisa, Dipartimento di Informatica (1999) TD-5/99.

[14] Gadducci, F., Miculan, M., Montanari, U.: About permutation algebras, (pre)sheaves and named sets. HOSC **19**(2-3) (2006) 283–304

[15] Fiore, M., Staton, S.: Comparing operational models of name-passing process calculi. IC **204**(4) (2006) 524–560

[16] Corradini, A., Heckel, R., Montanari, U.: Compositional sos and beyond: a coalgebraic view of open systems. TCS **280**(1-2) (2002) 163–192

[17] Ferrari, G.L., Montanari, U., Pistore, M.: Minimizing transition systems for name passing calculi: A co-algebraic formulation. In: FoSSaCS. (2002) 129–158

[18] Ferrari, G.L., Montanari, U., Tuosto, E.: Coalgebraic minimization of hd-automata for the pi-calculus using polymorphic types. TCS **331**(2-3) (2005) 325–365

[19] Caires, L., Cardelli, L.: A Spatial Logic for Concurrency (Part I). IC **186**(2) (2003) 194–235

[20] Caires, L.: Behavioral and spatial properties in a logic for the pi-calculus. In: FoSSaCS. (2004) 72–89

[21] Klin, B.: Coalgebraic modal logic beyond sets. In: MFPS. (2007) 177–201

[22] Goguen, J.A., Burstall, R.M.: Institutions: Abstract model theory for specification and programming. J. ACM **39**(1) (1992) 95–146

[23] Rydeheard, D.E., Burstall, R.M.: Computational category theory. Prentice Hall (1988)

[24] Bonsangue, M.M., Kurz, A.: Pi-calculus in logical form. In: LICS. (2007) 303–312

[25] Luks, E.M.: Permutation Groups and Polynomial Time Computation. DIMACS DMTCS **11** (1993) 139–175